IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT EXAMINING OPERATION

ATTN'Y DOCKET NO.: ETS-TCA

APPLICATION OF: PETER BRITTINGHAM, MARY E. MORLEY, MARK K.

SINGLEY, MARK G. ZELMAN, KRISHNA N. JHA, JAMES H. FIFE, ROBERT L. RARICH, IRVIN R.

KATZ, RANDY E. BENNETT

FOR: COMPUTER-BASED TEST-ITEM GENERATION AND

CLONING

VISUAL BASIC SOURCE CODE APPENDIX (VBSCA 1-469)

VISUAL BASIC SOURCE CODE APPENDIX TABLE OF CONTENTS¹

TCA.vbp VBSCA -1
AXProlog.vbp VBSCA -4
Common.bas
Main.bas VBSCA -6
modUtil.bas
MTAPI.BAS VBSCA -12
MTDeclaration.bas VBSCA -17
MTUtil.bas VBSCA -21
Timer.bas VBSCA -28
Contraint.frm
EditConstraint.frm VBSCA -50
Form1.frm
frmAbout.frm VBSCA -54
frmAttributes.frm VBSCA -55
frmComments.frm
frmDifficulty.frm VBSCA -62
frmDrag.frm VBSCA -76
frmIED.frm
frmIndexedString.frm

15. An this this the Car Call His Call

20

¹ All software COPYRIGHT 1999 ETS except for MTAPI.BAS

	frmNew.frm VBSCA -8
	frmNewModel.frm VBSCA -9
	frmProgram.frm VBSCA -9
5	frmProgress.frm
	frmProlog.frm
	frmSplash.frm VBSCA -10
	SetPrecision.frm
·	String.frm VBSCA -11
10	TCA.FRM VBSCA -11
100 mg/mg/ 100 mg/ 100 mg/	Variable.frm VBSCA -21
St. wasti mili	Application.cls VBSCA -25
ilian dan dan dan dan masa maja bilan bilan bilan bilan dan dan dan dan dan dan dan dan dan d	CClones.cls VBSCA -25
Trans.	CConstraints.cls VBSCA -26
	Checksum.cls VBSCA -26
ķ aš:	Clone.cls VBSCA -27
And And	CModels.cls VBSCA -27
	Constraint.cls VBSCA -28
	ConstraintSolver.cls
20	CVariables.cls VBSCA -29
	CVariants.cls VBSCA -30
	DifficultyEstimate.cls
	DocStatus.cls VBSCA -31
	DSMODEL.CLS

Family.cls	VBSCA -322-
File.cls	VBSCA -328-
FileFind.cls	VBSCA -333-
GMATDifficultyEstimate.cls	VBSCA -336-
GREDifficultyEstimate.cls	VBSCA -340-
IniFile.cls	VBSCA -345-
LockedItem.cls	VBSCA -350-
Model.cls	VBSCA -362-
PrintModel.cls	VBSCA -381-
Progress.cls	VBSCA -384-
Prolog.cls	VBSCA -386-
PSMODEL.cls	
QCModel.cls	VBSCA -403-
StringSolver.cls	VBSCA -410-
StringSolverx.cls	VBSCA -412-
SubString.cls	VBSCA -413-
Value.cls	VBSCA -417-
VarFraction.cls	VBSCA -419-
Variable.cls	VBSCA -428-
VarInteger.cls	VBSCA -432-
VarReal.cls	VBSCA -439-
VarString.cls	VBSCA -449-
VarI Intyned als	VRSCA -456-

Win32API.cls	VBSCA -462
Word.cls	VBSCA -466

	'TCA.vbp
	Type=Exe
	Reference=*\G{00020430-0000-0000-C000-000000000046}#2.0#0#\\.\.\WINNT\System32\
_	StdOle2.Tlb#OLE Automation
5	Reference=*\G{00020905-0000-0000-C000-00000000046}#8.0#409#\\.\Microsoft
	Office\Office\MSWORD8.OLB#Microsoft Word 8.0 Object Library
	Reference=*\G{953298D7-F0DE-11D2-AED3-00000000000}#13.0#0#AXProlog.exe#AXProl
	og
	Object={FE0065C0-1B7B-11CF-9D53-00AA003C9CB6}#1.1#0; COMCT232.OCX
10	Object={6B7E6392-850A-101B-AFC0-4210102A8DA7}#1.3#0; COMCTL32.OCX
	Object={BDC217C8-ED16-11CD-956C-0000C04E4C0A}#1.1#0; TABCTL32.OCX
	Object={F9043C88-F6F2-101A-A3C9-08002B2F49FB}#1.2#0; COMDLG32.OCX
	Form=TCA.frm
	Module=Util; modUtil.bas
15	Class=Model; Model.cls
	Class=Constraint; Constraint.cls
	Class=Variable, Variable.cls
	Class=TCAApplication; Application.cls
17 m m n n n n n n n n n n n n n n n n n	Module=StartUp; Main.bas
201	Form=Variable.frm
Ų"	Class=CVariables; CVariables.cls
zigu Em	Class=CConstraints; CConstraints.cls
M.	Form=Constraint.frm
	Class=MSWord; Word.cls
25	Form=frmSplash.frm
== ==================================	Class=VarInteger; VarInteger.cls
ui	Class=VarReal; VarReal.cls
	Class=VarFraction; VarFraction.cls
. .	Class=VarString; VarString.cls
- L	Form=frmIndexedString.frm
C)	Class=File; File.cls
	Class=CClones; CClones.cls
	Class=IniFile; IniFile.cls
2.5	Class=Win32API; Win32API.cls
35	Class=CModels; CModels.cls
	Class=Clone; Clone.cls
	Form=frmAttributes.frm
	Class=Family; Family.cls
40	Class=DocStatus; DocStatus.cls
40	Class=Checksum; Checksum.cls
	Form=frmProgress.frm
	Class=Progress; Progress.cls
	Form=frmDifficulty.frm
	Class=DifficultyEstimate; DifficultyEstimate.cls

Class=SMCModel; PSModel.cls Class=OCModel; qcmodel.cls Class=DSModel; dsmodel.cls Class=VarUntyped; VarUntyped.cls 5 Class=LockedItem; LockedItem.cls Class=GMATDifficultyEstimate; GMATDifficultyEstimate.cls Form=frmAbout.frm Form=frmNew.frm 10 Form=String.frm Class=SubString; SubString.cls Class=ConstraintSolver; ConstraintSolver.cls Class=StringSolver; StringSolver.cls Class=Value; Value.cls 15 Class=PrintModel: PrintModel.cls Module=MTAPI; MTAPI.bas Module=MTDeclarations; MTDeclarations.bas Module=MTUtil; MTUtil.bas Form=frmProlog.frm 20 ResFile32="Tca.res" IconForm="frmTCA" ďì Hay they had Startup="Sub Main" HelpFile="" Title="TCA" ExeName32="TCA.exe" Command32="" Name="Project1" HelpContextID="0" CompatibleMode="0" MajorVer=0 MinorVer=1 RevisionVer=145 AutoIncrementVer=1 ServerSupportFiles=0 35 VersionCompanyName="ETS" CompilationType=0 OptimizationType=2 FavorPentiumPro(tm)=0 CodeViewDebugInfo=0 40 NoAliasing=0 BoundsCheck=0 OverflowCheck=0 FlPointCheck=0 FDIVCheck=0

45

UnroundedFP=0

Class=GREDifficultyEstimate; GREDifficultyEstimate.cls

StartMode=0
Unattended=0
Retained=0
ThreadPerObject=0

MaxNumberOfThreads=1

Type=OleExe Reference=*\G{00020430-0000-0000-C000-00000000046}#2.0#0#..\..\..\WINNT\System32\ STDOLE2.TLB#OLE Automation 5 Reference=*\G{3D5C6BF0-69A3-11D0-B393-00A0C9055D8E}#1.0#0#..\..\Common Files\designer\MSDERUN.DLL#Microsoft Data Environment Instance 1.0 Reference=*\G{00000200-0000-0010-8000-00AA006D2EA4}#2.0#0#..\..\Common Files\system\ado\msado20.tlb#Microsoft ActiveX Data Objects 2.0 Library Class=Prolog; Prolog.cls Module=Module1; Timer.bas 10 Class=File; File.cls Startup="(None)" HelpFile="" ExeName32="AXProlog.exe" 15 Command32="" Name="AXProlog" HelpContextID="0" CompatibleMode="1" CompatibleEXE32="AXProlog.exe" ď) MajorVer=1 20 Ų" MinorVer=0 RevisionVer=0 Man Hone AutoIncrementVer=0 ServerSupportFiles=0 25 VersionCompanyName="ETS" CompilationType=0 OptimizationType=0 FavorPentiumPro(tm)=0 CodeViewDebugInfo=0 NoAliasing=0 BoundsCheck=0 OverflowCheck=0 FlPointCheck=0 FDIVCheck=0 35 UnroundedFP=0 StartMode=1 Unattended=-1 Retained=0 ThreadPerObject=-1

' AXProlog.vbp

40

MaxNumberOfThreads=1 DebugStartupOption=0

' Common.bas Attribute VB_Name = "Common"

```
' Main.bas
       Attribute VB_Name = "StartUp"
       Option Explicit
       Public Const READ UNTIL EOF = 0
      Public Const INI DIRECTORY = "C:\TCS\TCA\OUT\TCAOUT.INI"
 5
      Public Const IN DIRECTORY = "C:\TCS\TCA\IN\"
      Public Const OUT DIRECTORY = "C:\TCS\TCA\OUT\"
       Public Const LOCKED ITEM NAME = "TCATEMP.DOC"
      Public Const LVM_FIRST = &H1000
      Public Const LVM SETEXTENDEDLISTVIEWSTYLE = LVM FIRST + 54
10
      Public Const LVM GETEXTENDEDLISTVIEWSTYLE = LVM FIRST + 55
       Public Const LVS EX FULLROWSELECT = &H20
      Public Const HALT_FN = "C:\HALT.TCA"
       Public Const STRING DELIMITER = 164
151
      Private Sub Main()
 ű
Dim MyApp As New TCAApplication
        If App.PrevInstance Then
          Call MsgBox("Only one instance of TCA may be run at a time!",
             vbExclamation, "Error")
          Exit Sub
        End If
         ' 10 seconds for component timeout
         App.OleRequestPendingTimeout = 10000
25
        MyApp.Run
```

End Sub

```
Attribute VB Name = "Util"
        Option Explicit
        'Capitalizes the first letter of a string if it's a lower case letter
        Sub CapitalizeString(strInput As String)
 5
          Dim str1, str2 As String
          Dim intStrLen As Integer
          intStrLen = Len(strInput)
          If (intStrLen > 0) Then
10
             str1 = UCase(left(strInput, 1))
          End If
          If (intStrLen > 1) Then
             str2 = right(strInput, intStrLen - 1)
          End If
 T)
 Œ١
15
          strInput = str1 \& str2
        End Sub
        'Selects contents of text box for easy editing
        Sub txtSelectAll(txtTextBox As TextBox)
          ' Automatically select all text
          txtTextBox.SelStart = 0
          txtTextBox.SelLength = Len(txtTextBox.Text)
        End Sub
        'Checks to see if a file exists
        Function FileExists(ByVal strFN As String) As Boolean
          Dim intFNum As Integer
25
          'Get the file number
          intFNum = FreeFile
          'Open the file and trap any errors
          On Error GoTo NotFound
```

' modUtil.bas

```
Open strFN For Binary Access Read As #intFNum
          On Error GoTo 0
          Close #intFNum
          FileExists = True
 5
          Exit Function
        NotFound:
          'Close the file
          Close #intFNum
          FileExists = False
          Exit Function
10
        End Function
        'extracts the path from a path/filename string
 Mr. Th
        Function ExtractPath(ByVal strFN As String) As String
 m
 U
          Dim varI1 As Variant
Dim varI2 As Variant
          ' find the last "\" in the string
          varI1 = 0
          Do
            varI2 = varI1
            varI1 = InStr(varI2 + 1, strFN, "\")
          Loop Until varI1 = 0
          ExtractPath = Mid(strFN, 1, varI2)
        End Function
        'extracts the file name from a path/filename string
        Function ExtractFileName(ByVal strFN As String) As String
25
          Dim varI1 As Variant
          Dim varI2 As Variant
          ' find the last "\" in the string
          varI1 = 0
30
          Do
```

```
varI2 = varI1
            varI1 = InStr(varI2 + 1, strFN, "\")
          Loop Until varI1 = 0
          ExtractFileName = Mid(strFN, varI2 + 1, Len(strFN) - varI2)
 5
        End Function
        'extracts the file name sans extension from a path/filename string
        Function ExtractFileNameNoExt(ByVal strFN As String) As String
          strFN = ExtractFileName(strFN)
          Dim varI1 As Variant
10
          Dim varI2 As Variant
          ' find the last "." in the string
          varI1 = 0
          Dο
            varI2 = varI1
 d
            varI1 = InStr(varI2 + 1, strFN, ".")
157
 U1
          Loop Until varI1 = 0
          ExtractFileNameNoExt = Mid(strFN, 1, varI2 - 1)
        End Function
        ' extracts the family name - everything up to $R
        Function ExtractFamilyName(ByVal strFN As String) As String
          strFN = ExtractFileName(strFN)
 Cj
          Dim varI As Variant
          ' find "$R" in the string
          varI = InStr(1, strFN, "$R")
25
          If varI > 0 Then
            ExtractFamilyName = Mid(strFN, 1, varI - 1)
          End If
        End Function
30
        'extracts the key, meaning $R and everthing up to the.
        Function ExtractFamilyKey(ByVal strFN As String) As String
```

```
strFN = ExtractFileName(strFN)
           Dim varI As Variant
           Dim varI1 As Variant
           Dim varI2 As Variant
 5
           ' find "$R" in the string
           varI = InStr(1, strFN, "$R")
          ' find the last "." in the string
           varI1 = 0
           Do
10
             varI2 = varI1
             varI1 = InStr(varI2 + 1, strFN, ".")
           Loop Until varI1 = 0
           ExtractFamilyKey = Mid(strFN, varI, varI2 - varI)
        End Function
 4)
15
        ' trim nulls off the end of a string
Function TrimAtFirstNull(ByVal strS As String) As String
           Dim varI As Variant
           varI = InStr(1, strS, Chr(0))
           TrimAtFirstNull = left(strS, varI - 1)
        End Function
        ' returns a string with all instances of strFrom replaced
        ' with strTo in string strS
        Function ReplaceAll(ByVal strS As String, ByVal strFrom As String, _
           ByVal strTo As String) As String
25
          Dim varI As Variant
           Dim intL As Integer
          Do
             varI = InStr(1, strS, strFrom)
             If varI > 0 Then ' found strFrom
30
               intL = Len(strS)
               strS = left(strS, varI - 1) & strTo & _
                  right(strS, intL - Len(strFrom) - varI + 1)
             End If
```

```
ReplaceAll = strS
        End Function
        'returns the name of indexed string variables
        Function GetIndexedName(ByVal strName As String, _
 5
          ByVal intI As Integer) As String
          GetIndexedName = strName & "." & Trim(Str(intI))
        End Function
10
        ' Prolog shuts down when this file is created
        Sub CreateKillFile()
          Open HALT_FN For Output As #10
          Print #10, "Halt!"
          Close #10
 ij.
 đ١
15
        End Sub
 Min allen Men aben
        'Delete the kill file
        Sub DestroyKillFile()
          On Error Resume Next ' if it's not there, Kill will produce an error
          Kill HALT FN
          err.Clear
        End Sub
 C)
```

Loop Until varI = 0

```
Attribute VB Name = "MTAPI"
        'mtapi.bas 4.0
 5
        '(c) Copyright 1992-1999 by Design Science, Inc. All rights reserved
        ' with the exception that registered MathType owners may alter these
       ' macros for use by themselves and other registered MathType owners
        ' provided that:
          1) The alterations are summarized in a comment directly below this
            copyright notice. The comment should start with the words
10
            "Modified by" and include the name of the person altering the
           macros, the date of alteration, and that person's email address
            (if available).
          2) Persons altering the macros notify Design Science of the nature
            of any changes they have made.
15
        'These provisions may help us help other customers, and will help us
       'continue to provide quality products for you in the future.
 ij,
       'version # of this API
 g1
201
       Public Const MTAPI VERSION = 4
 Man den Sen den
       ' maximum length of file paths, names, etc.
       Public Const MTAPI MAX PATH = 260
       ' Picture specifier
       Public Type MTAPI PICT
         mm As Long
         xExt As Long
         yExt As Long
         hMF As Long
       End Type
30
       Public Type RECT
         left As Long
                As Long
         top
         right As Long
         bottom As Long
35
       End Type
       ' Picture dimensions
       Public Type MTAPI DIMS
         baseline As Integer 'dist of baseline from bottom (points)
         bounds As RECT
                              'bounding rectangle (points)
```

'MTAPI.BAS

End Type

' return codes from MT DLL API

' success, no error

Public Const mtOK = 0

5 ' equation OLE 1.0 object on clipboard

Public Const mtOLE EQUATION = 1

'Windows metafile equation graphic (not OLE object) on clipboard

Public Const mtWMF EQUATION = 2

'Macintosh PICT equation graphic (not OLE object) on clipboard

Public Const mtMAC_PICT_EQUATION = 4 10

' equation OLE 2.0 object on clipboard

Public Const mtOLE2 EQUATION = 8

' can't find MathType application

Public Const mtMT_NOT_FOUND = -1 151

' can't run the MathType application

Han Car Car Public Const mtMT CANT RUN = -2

' the MathType application is the wrong version

Public Const mtMT_BAD_VERSION = -3

2**0** ' the MathType application is already in use

Public Const mtMT IN USE = -4

' the MathType application is not running (i.e. unexpectedly aborted)

Public Const mtMT_NOT_RUNNING = -5

' time ran out waiting for the MathType application to start up

Public Const mtRUN_TIMEOUT = -6

' not equation on clipboard ļa h

Public Const mtNOT EQUATION = -7

' file does not exist or bad pathname

Public Const mtFILE_NOT_FOUND = -8

30 ' insufficient memory

Public Const mtMEMORY = -9

'bad file

Public Const mtBAD_FILE = -10

' requested data does not exist

Public Const mtDATA NOT FOUND = -11 35

' too many server session open

Public Const mtTOO MANY SESSIONS = -12

' could not perform one or more subs

Public Const mtSUBSTITUTION ERROR = -13

' could not perform translation 40

Public Const mtTRANSLATOR ERROR = -14

^{&#}x27; error return codes

' could not set preferences, or invalid preference string Public Const mtPREFERENCE ERROR = -15 ' other error Public Const mtERROR = -9999 'options values for MTInitAPI Public Const mtinitLAUNCH_AS_NEEDED = 0 Public Const mtinitLAUNCH NOW = 1 'options values for MTGetTranslatorsInfo Public Const mttrnCOUNT = 1Public Const mtrnMAX NAME = 2Public Const mttrnMAX DESC = 3 Public Const mttrnMAX FILE = 4 Public Const mttrnOPTIONS = 5'options values for MTXFormAddVarSub Public Const mtxfmSUBST ALL = 0Public Const mtxfmSUBST ONE = 1 'find/replace types for MTXFormAddVarSub substitutions Public Const mtxfmVAR SUB BAD = -1 Public Const mtxfmVAR SUB PLAIN TEXT = 0 Public Const mtxfmVAR SUB MTEF TEXT = 1 Public Const mtxfmVAR SUB MTEF_BINARY = 2 Public Const mtxfmVAR SUB DELETE = 3 'replace style for MTXFormAddVarSub substitutions when replaceType = mtxfmVAR SUB PLAIN_TEXT Public Const mtxfmSTYLE TEXT = 1 Public Const mtxfmSTYLE FUNCTION = 2 Public Const mtxfmSTYLE_VARIABLE = 3 Public Const mtxfmSTYLE LCGREEK = 4 Public Const mtxfmSTYLE UCGREEK = 5 Public Const mtxfmSTYLE SYMBOL = 6 Public Const mtxfmSTYLE VECTOR = 7 Public Const mtxfmSTYLE_NUMBER = 8 ' options values for MTXFormSetPrefs Public Const mtxfmPREF EXISTING = 1 Public Const mtxfmPREF MTDEFAULT = 2 Public Const mtxfmPREF USER = 3 Public Const mtxfmPREF LAST = 3

'options values for MTXFormSetTranslator

5

10

15

ij

gn Un

2**0**

30

35

```
Public Const mtxfmTRANSL INC NONE = 0
       Public Const mtxfmTRANSL INC NAME = 1
       Public Const mtxfmTRANSL INC DATA = 2
       Public Const mtxfmTRANSL INC MTDEFAULT = 4
 5
       'return values from MTXFormGetStatus
       Public Const mtxfmSTAT PREF = -3
       Public Const mtxfmSTAT_TRANSL = -2
       Public Const mtxfmSTAT ACTUAL LEN = -1
       ' data sources/destinations for MTXFormEqn
       Public Const mtxfmPREVIOUS = -1
10
       Public Const mtxfmCLIPBOARD = -2
       Public Const mtxfmLOCAL = -3
       ' data formats for MTXFormEqn
       Public Const mtxfmMTEF = 4
15
       Public Const mtxfmHMTEF = 5
       Public Const mtxfmPICT = 6
       Public Const mtxfmTEXT = 7
 ű
 <u>M</u>
       Public Const mtxfmHTEXT = 8
 Uĩ
 'option values for MTSetMTPrefs
20
       Public Const mtprfMODE NEXT_EQN = 1
       Public Const mtprfMODE MTDEFAULT = 2
       Public Const mtprfMODE INLINE = 4
       'MT API functions
       Public Declare Function MTAPIVersion Lib "mt4" (ByVal api As Integer) As Long
       Public Declare Function MTInitAPI Lib "mt4" (ByVal options As Integer, ByVal timeout As
       Integer) As Long
       Public Declare Function MTTermAPI Lib "mt4" () As Long
       Public Declare Function MTClearClipboard Lib "mt4" () As Long
       Public Declare Function MTEquationOnClipboard Lib "mt4" () As Long
       Public Declare Function MTXFormReset Lib "mt4" () As Long
30
       Public Declare Function MTXFormAddVarSub Lib "mt4" (
         ByVal options As Integer, _
         ByVal findType As Integer, ByVal find As String, ByVal findLen As Long,
         ByVal replaceType As Integer, ByVal replace As String, ByVal replaceLen As Long,
         ByVal replaceStyle As Integer
35
       ) As Long
       Public Declare Function MTXFormSetTranslator Lib "mt4" (ByVal options As Integer, _
         ByVal transName As String) As Long
       Public Declare Function MTXFormSetPrefs Lib "mt4" (ByVal prefType As Integer, ByVal
40
       prefStr As String) As Long
```

Public Declare Function MTSetMTPrefs Lib "mt4" (ByVal mode As Integer, ByVal prefs As String, _

ByVal timeout As Integer) As Long

Public Declare Function MTXFormEqn Lib "mt4" (_

- ByVal src As Integer, ByVal srcFmt As Integer, ByVal srcData As String, ByVal srcDataLen As Long,
 - ByVal dst As Integer, ByVal dstFmt As Integer, ByVal dstData As String, ByVal dstDataLen As Long, _
 - ByRef dims As MTAPI_DIMS) As Long
- 10 Public Declare Function MTXFormGetStatus Lib "mt4" (ByVal index As Integer) As Long

'Windows API declarations
Public Declare Function WinHelp Lib "user32" Alias "WinHelpA" (ByVal hwnd As Long, ByVal lpHelpFile As String, ByVal wCommand As Long, ByVal dwData As Long) As Long Public Declare Function LoadLibrary Lib "kernel32" Alias "LoadLibraryA" (ByVal lpLibFileName As String) As Long
Public Declare Function FreeLibrary Lib "kernel32" (ByVal hLibModule As Long) As Long Public Declare Function LoadString Lib "user32" Alias "LoadStringA" (ByVal hInstance A Long, ByVal wID As Long, ByVal lpBuffer As String, ByVal nBufferMax As Long) As Lo Public Declare Function GetLocaleInfo Lib "kernel32" Alias "GetLocaleInfoA" (ByVal LocaleInfo As Long, ByVal LCType As Long, ByVal lpLCData As String, ByVal cchData As Long) A Long
Public Declare Function GetEnvironmentVariable Lib "kernel32" Alias "GetEnvironmentVariableA" (ByVal lpName As String, ByVal lpBuffer As String, ByVal r As Long) As Long
Public Declare Function SetEnvironmentVariable Lib "kernel32" Alias "SetEnvironmentVariableA" (ByVal lpName As String, ByVal lpValue As String) As Long Public Declare Function GetTickCount Lib "kernel32" () As Long
'' 'Constants for use in Windows API calls
' 'values for LCType (locale info requested) - used in MTLib.InitLocaleStrs Public Const Locale_SLanguage As Long = &H2 Public Const Locale_SEngLanguage As Long = &H1001
' 'Constants for use in Help calls
Public Const hlpMSWDPreferences_Dialog = 117 Public Const hlpMSWDEquation_Number_Format_Dialog = 6300 Public Const hlpMSWDFormat Equations_Dialog = 6500
Public Const hlpMSWDInsert_Equation_Section_Dialog = 114 Public Const hlpMSWDFormat_Equation_Section_Dialog = 116 Public Const hlpMSWDSet_Equation_Preferences_Dialog = 37 Public Const hlpMSWDConvert_Equations_Dialog = 44
Public Const hlpMSWDInsert_Equation_Number_Dialog = 118 Public Const hlpMSWDInsert_Requation_Ref_Dialog = 119
Public Const hlpMSWDWT_SetEqnPrefs = 122

	Public Const hlpMSWDWT_InlineEqn = 123 Public Const hlpMSWDWT_CenteredEqn = 124 Public Const hlpMSWDWT_CenteredNumberedEqn = 125 Public Const hlpMSWDWT_EqnNumber = 126
5	Public Const hlpMSWDWT_EqnRef = 127 Public Const hlpMSWDWT_EqnSec = 128 Public Const hlpMSWDWT_ModEqnSec = 129 Public Const hlpMSWDWT_FormatEqnNum = 130
10	Public Const hlpMSWDWT_ConvertEqn = 131 Public Const hlpMSWDWT_FormatEqn = 132 Public Const hlpMSWDWT_UpdateEqn = 133
	Constants for use in the MathType Commands
15	' Public Const mtversMajVerHi = 1279 '0x04ff
	Public Const mtversMajVerLo = 1024 '0x0400 Public Const mtversMinVer = 1024 '0x0400
2 <u>4</u>	' Public Const mtreg_MT_LANG_LOCATION As String = "HKEY_CURRENT_USER\Software\Design Science\DSMT4\Config" 'Registry entry for
To a control of the c	MathType's curent language Public Const mtreg_MT_LANG_KEY As String = "AppLang" 'registry key for MathType's curent language
	Public Const mtreg_MT_PROGDIR_LOCATION As String = "HKEY_LOCAL_MACHINE\SOFTWARE\Design Science\DSMT4\Directories" 'Registry entry for MathType's directory Public Const mtreg MT_PROGDIR_KEY As String = "ProgDir" 'registry key for MathType's
Ç.	directory
3 0 ₁	Public Const mtreg_MT_LANGUAGEDIR_LOCATION As String = "HKEY_LOCAL_MACHINE\SOFTWARE\Design Science\DSMT4\Directories" 'Registry ontagger for MothType's language support files directory
	entry for MathType's language support files directory Public Const mtreg_MT_LANGUAGEDIR_KEY As String = "LastLangDir" 'registry key for MathType's language support files directory
35	Public Const mtreg_MT_HELPDIR_LOCATION As String = "HKEY_LOCAL_MACHINE\SOFTWARE\Design Science\DSMT4\Directories" 'Registry entry for MathType's help file directory Public Const mtreg_MT_HELPDIR_KEY As String = "LastHelpDir" 'registry key for
40	MathType's help file directory Public Const mtreg_MT_HELPFILE_LOCATION As String = "HKEY_CURRENT_USER\Software\Design Science\DSMT4\Config" 'Registry entry for MathType's help file name Public Const mtreg_MT_HELPFILE_KEY As String = "HelpFile" 'registry key for
	MathType's help file name

	Public Const mtreg_MT_SYSTEMDIR_LOCATION As String = "HKEY_LOCAL_MACHINE\SOFTWARE\Design Science\DSMT4\Directories" 'Registry
	entry for MathType's system directory
	Public Const mtreg_MT_SYSTEMDIR_KEY As String = "LastAppSystemDir" 'registry key
5	for MathType's system directory
	Public Const mtreg MT_PREFDIR_LOCATION As String =
	"HKEY_LOCAL_MACHINE\SOFTWARE\Design Science\DSMT4\Directories" 'Registry
	entry for MathType's preferences folder
	Public Const mtreg_MT_PREFDIR_KEY As String = "LastPrefsDir" 'registry key for
10	MathType's system directory
	Public Const mtreg_MT_WORDCMDS_LOCATION As String =
	"HKEY_CURRENT_USER\SOFTWARE\Design Science\DSMT4\WordCommands"
	'Registry entry for MathType's Word Commands data
	Public Const mtreg_MT_WORD_CONVFROM As String = "ConvertFrom" 'ConvertFrom
15	key
	Public Const mtreg_MT_WORD_CONVTO As String = "ConvertTo" 'ConvertTo key
	Public Const mtreg_MT_WORD_CONVMISC As String = "ConvertMisc" 'ConvertMisc key
	Public Const mtreg_MT_WORD_CONVTRANS As String = "ConvertTranslator"
20°	'ConvertTranslator key
201	Public Const mtreg_MT_WORD_DONTSHOW_EQNREFDLG As String =
FA E	"NoInsertEqnRefDlg" 'Don't Show Insert Eqn Ref dialog key
	Public Const mtreg_MT_WORD_DONTSHOW_SLOWEQNUPDATE As String =
કરિકા તીમા કરિકા મીમા ઉત્તર માત્ર મુખા છે.	"NoSlowUpdateEqnDlg" 'Don't Show Insert Eqn Ref dialog key
	Public Const mtreg_MT_WORD_DONTSHOW_LANGDLLERROR As String =
253 11 11 11	"NoLanuageDLLError" 'Don't show Missing Lang DLL error key
ļ.	'Strings used in MT text equations (TeX and MathML)
C)	Public Const mttexteqn_START As String = "% MathType!" 'The identifier at the beginning
C)	of MathType translator text equations Public Const methods and String = "9/ MathType End!" The identifier at the end of
20	Public Const mttexteqn_END As String = "% MathType!End!" 'The identifier at the end of MathType translator text equations
30	Matiri ype translator text equations
	' Property names
	Public Const mtprop_USE_MATHTYPE_PREFS As String = "MTUseMTPrefs" 'The
	name of the Document Property that indicates to use MathType's prefs for new equations
	Public Const mtprop_PREFERENCES As String = "MTPreferences" 'Contains the
35	doc's settings for new equations
	Public Const mtprop_PREFERENCES_FILE As String = "MTPreferenceSource" 'Contains
	the doc's settings for new equations Public Const mtprop NUMBER PREFS As String = "MTEquationNumber" 'Contains
	Public Const mtprop_NUMBER_PREFS As String = "MTEquationNumber" 'Contains the current equation number format preferences
40	Public Const mtprop_DEFER_FIELD_UPDATE As String = "MTDeferFieldUpdate"
4 0	I done Coust inchrob Detret Treep Ordate As String - MIDeterried obtains

'Controls field updating Public Const mtpropEQUATION_SECTION_CHECKED As String = "MTEquationSection" 'Indicates if eqn section number is 0 check has been made Public Const mtprop_EQNREFPANE As String = "MTEqnRefPane" 'Pane number containing insertion point where ref. is to be placed 5 '----- AutoText entry names -----Public Const mtautotext MT3 EQN NUMBER FORMAT As String = "ZMTEqnNumFormatPrefs" 'The name of old Autotext entry that held MathType3's equation number format prototype 10 '----- MathType OLE data -----Public Const mtole PROGID As String = "Equation.DSMT4" 'OLE Prog ID used to identify MathType 4 '----- Style names -----Public Const mtstyle EQUATION_SECTION As String = "MTEquationSection" 'Style used for eqn. section names 15 Public Const mtstyle_DISPLAY_EQUATION As String = "MTDisplayEquation" 'Style used for display equations ų] D1 Uī '----- Misc. constants -----'Constants used to specify 'curent selection' or 'whole document' 2**0**] Public Const mt RANGE DOCUMENT = 0 Public Const mt RANGE SELECTION = 1 'Constants used by MTMsgBox Public Const mt MBYESNO = 1 Public Const mt MBYESNOCANCEL = 2 Public Const mt MBYES = 1Public Const mt MBNO = 2Public Const mt MBCANCEL = 3 'Flag bit for MTLib.SaveWordState() Public Const mt SWS TRACKCHANGES = 1 Public Const mt SWS SMART CUTPASTE = 2 30

Public Const mt SWS TYPING_REPLACE_SELECTION = 4

```
If MTInitAPI(mtinitLAUNCH AS NEEDED, 30) 

◆ 0 Then
             msg$ = MTUtil.GetUserString("!1606The MathType commands could not communicate
        with MathType. There was a problem starting the API. Please be sure that MathType is properly
        installed.")
            CheckMTDLLVersion = 0
 5
            errorflag = 1
          Else
             'get the API Version
            dllver = MTAPIVersion(MTAPI VERSION)
10
            'check the version against our constants
            If (dllver > mtversMajVerHi) Or (dllver < mtversMajVerLo) Then
               msg$ = MTUtil.GetUserString("!1607The version of this macro doesn't match the
        version of MathType's DLL. Reinstall MathType to fix this condition.")
               CheckMTDLLVersion = 0
15
               errorflag = 1
            ElseIf (dllver < mtversMinVer) Then
               msg$ = MTUtil.GetUserString("!1608A more recent version of MathType's DLL is
        required to use this macro. Reinstall MathType to fix this condition.")
               CheckMTDLLVersion = 0
 ij,
               errorflag = 1
201
 Me Man Wen American
            End If
          End If
          If (errorflag = 1) Then
                                  'report error condition
            MsgBox msg$, vbCritical, MTUtil.GetUserString("!1609MathType Commands for
        Microsoft Word Error")
          End If
        End Function
 F£.
 C
 C
                   GetUserString$
30
        Public Function GetUserString$(EnglishString$)
          'simply return the English version (strip "!nnnn" from start)
          GetUserString$ = right(EnglishString$, Len(EnglishString$) - 5)
        End Function
35
                       GetMathTypeDir$
          Gets the location of MathType from the registry
       Public Function GetMathTypeDir$()
40
          Dim path$
```

'get the location of Mathtype from the registry path\$ = System.PrivateProfileString("", mtreg_MT_PROGDIR_LOCATION, mtreg MT PROGDIR KEY) 'return the results 5 GetMathTypeDir\$ = path\$ **End Function** WritePermSetting 'Writes key/value pair to permanent location, ie Windows registry. 10 'Used when data needs to be saved whose scope is larger than a document. Public Sub WritePermSetting(key\$, data\$) System.PrivateProfileString("", mtreg_MT_WORDCMDS_LOCATION, key\$) = data\$ End Sub ReadPermSetting\$ G Li 'Reads key's value from the permanent location, ie Windows registry. 'Used when data needs to be saved whose scope is larger than a document. 2**0** Public Function ReadPermSetting\$(key\$) ReadPermSetting\$ = System.PrivateProfileString("", mtreg MT WORDCMDS LOCATION, key\$) **End Function** SetNextTXFormPrefs 'Sets prefs that MathType will use for the next transformed equation. 'Returns MTXFormSetPrefs result code. Function SetNextTXFormPrefs(prefStr\$) Dim stat 30 'set preferences for next transformed equation stat = MTXFormSetPrefs(mtxfmPREF USER, prefStr\$) If stat \Leftrightarrow 0 Then MsgBox MTUtil.GetUserString("!1100There was a problem sending your equation 35 preferences for " + "this document to MathType. This equation will use MathType's " + "'New Equation' preferences."), vbExclamation, MTUtil.GetUserString("!1101MathType Preferences Problem") End If

```
End Function
                   SetPrefsForNextEqn
        'Sets prefs that MathType will use for the next new equation.
 5
        'Returns MTSetMTPrefs result code.
        Public Function SetPrefsForNextEqn(prefStr$, inline As Boolean)
          Dim stat
10
          Dim options As Integer
          options = mtprfMODE NEXT EQN
          If inline Then options = options + mtprfMODE INLINE
          'set preferences for next transformed equation
          stat = MTSetMTPrefs(options, prefStr$, -1)
          If stat <> 0 Then
15
            MsgBox MTUtil.GetUserString("!1100There was a problem sending your equation
        preferences for "
               + "this document to MathType. This equation will use MathType's "
 ű
              + "'New Equation' preferences."), vbExclamation, _
 ٥ì
              MTUtil.GetUserString("!1101MathType Preferences Problem")
20
 100 mm
          End If
          SetPrefsForNextEqn = stat
        End Function
                   IsEquationProgID
        'Returns 1 if the progID is a MathType/EE OLE1 progID.
 ų.
        'Returns 2 if the progID is a MathType/EE OLE2 progID.
        'Returns 0 if not a recognized progID.
 ļ.
        Public Function IsEquationProgID(progID$) As Long
301
          Dim uProgID$
          uProgID$ = UCase(progID$)
          If uProgID$ = "EQUATION" Then
            IsEquationProgID = 1
35
          ElseIf InStr(1, uProgID$, "EQUATION.", vbBinaryCompare) = 1 Then
            IsEquationProgID = 2
          Else
            IsEquationProgID = 0
          End If
        End Function
40
```

SetNextTXFormPrefs = stat

5	'Attempts to transform the graphic on the clipboard into an equation. 'Resulting format depends on how MathType has been configured by a 'previous call to MTXFormSetTranslator. 'The transformed equation is left on the clipboard. 'If OK, returns mtOK 'If not an equation, or an error occurred, returns mtNOT_EQUATION
10	Public Function TransformGraphicEquation() As Long TransformGraphicEquation = mtNOT_EQUATION
	'Use API call to check clipboard contents first If MTEquationOnClipboard() = mtNOT_EQUATION Then Exit Function End If
15	TransformGraphicEquation = TransformEquation() End Function
25.4.34.4.4.4.4.4.4.4.4.4.4.4.4.4.4.4.4.4	'Attempts to transform the item on the clipboard into an equation. 'Resulting format depends on how MathType has been configured by a 'previous call to MTXFormSetTranslator. 'The transformed equation is left on the clipboard. 'If OK, returns mtOK 'If not an equation, or an error occurred, returns mtNOT_EQUATION
	Public Function TransformEquation() As Long Dim stat As Long Dim dummyStr1\$, dummyStr2\$ Dim dummyDims As MTAPI_DIMS
30	On Error GoTo err
	stat = mtNOT_EQUATION
35	'as long as everything's OK, update the equation 'set aside some buffers dummyStr1\$ = Space(1) dummyStr2\$ = Space(1) With dummyDims .baseline = 0 .bounds.bottom = 0 .bounds.left = 0
40	hounds right = 0

```
.bounds.top = 0
         End With
         'do the update
          stat = MTXFormEqn(mtxfmCLIPBOARD, mtxfmTEXT, dummyStr1$, 1,
            mtxfmCLIPBOARD, mtxfmTEXT, dummyStr2$, 1, dummyDims)
 5
         If stat < 0 Then
            stat = mtNOT\_EQUATION
         End If
          GoTo Bye
10
         If err.Number = 5690 Or err.Number = 4198 Then
            'the user has revisions on, and this is an old revision that has been deleted
            stat = -2
            Resume Bye
15
         Else
            err.Raise err.Number
            Stop
         End If
       Bye:
TransformEquation = stat
       End Function
                  DeleteDocProperty
       'deletes document property, OK to call if it doesn't exist
       Public Function DeleteDocProperty(doc As Document, prop$)
         On Error GoTo Error
         doc.CustomDocumentProperties(prop$).Delete
       Error:
       End Function
30
                  DocPropertyExists
       'returns True if the active document contains the custom doc property
       Public Function DocPropertyExists(propName$) As Boolean
35
         Dim name$
         DocPropertyExists = False
         On Error GoTo Error
         name$ = ActiveDocument.CustomDocumentProperties(propName$).name
```

Error: **End Function** 5 Delay 'Pauses execution for timeout (in milliSecs) Public Sub Delay(timeout As Long) Dim start As Long 10 start = GetTickCount() Do While (GetTickCount() < (start + timeout)) DoEvents 'Yield to other processes. Loop End Sub 15

DocPropertyExists = True

```
Attribute VB Name = "Module1"
       Option Explicit
       Declare Function SetTimer Lib "user32" (ByVal hWnd As Long, _
         ByVal nIDEvent As Long, ByVal uElapse As Long, ByVal lpTimerProc As Long)
 5
          As Long
       Declare Function KillTimer Lib "user32" (ByVal hWnd As Long, _
          ByVal nIDEvent As Long) As Long
10
       Public gProlog As Prolog
       Public gTimerID As Long
        ' called by SolveConstraintsRandomly in Prolog.cls
       Public Sub SolveAsync()
          ' calls TimerCallback when timer runs out (it's set for 0, so it
          'runs out immediately. TimerCallback, and anything called by
The man wife will water with
         'TimerCallback, run async.
          gTimerID = SetTimer(0, 0, 1000, AddressOf TimerCallback)
       End Sub
       Public Sub TimerCallback(ByVal hWnd As Long, ByVal uMsg As Long, ByVal idEvent As
       Long, ByVal dwTime As Long)
          KillTimer 0, gTimerID
          gProlog.SolveConstraintsAsync ' in Prolog.cls
```

'Timer.bas

End Sub

```
'Contraint.frm
       VERSION 5.00
       Object = "{BDC217C8-ED16-11CD-956C-0000C04E4C0A}#1.1#0"; "TABCTL32.OCX"
       Begin VB.Form frmConstraints
 5
        BorderStyle = 4 'Fixed ToolWindow
                    = "Create or Change Constraints"
        Caption
        ClientHeight = 6405
        ClientLeft
                    = 45
        ClientTop
                     = 285
10
        ClientWidth = 6285
                     = "Form1"
        LinkTopic
                      = 0 'False
        MaxButton
                     = 0 'False
        MinButton
                     = 6405
        ScaleHeight
        ScaleWidth
15
                     = 6285
        ShowInTaskbar = 0 'False
        StartUpPosition = 1 'CenterOwner
        Begin TabDlg.SSTab sstConstraintTool
          Height
                     = 3375
          Left
                    = 240
 Ľĩ
          TabIndex
                      = 5
 May May Ales
                    = 1080
          Top
          Width
                     = 4455
          ExtentX
                      = 7858
25
          ExtentY
                      = 5953
                      = 393216
           Version
          TabHeight
                       = 520
          BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
                       = "MS Sans Serif"
            Name
                      = 8.25
            Size
                       = 0
            Charset
                       = 400
            Weight
            Underline
                        = 0 'False
            Italic
                     = 0 'False
            Strikethrough = 0 'False
35
          EndProperty
          TabCaption(0) = "Operators"
          TabPicture(0) = "Constraint.frx":0000
          Tab(0).ControlEnabled= -1 'True
40
          Tab(0).Control(0)= "cmdElseIf"
          Tab(0).Control(0).Enabled= 0 'False
          Tab(0).Control(1)= "cmdElse"
          Tab(0).Control(1).Enabled= 0 'False
          Tab(0).Control(2)= "cmdThen"
```

	Tab(0).Control(2).Enabled= 0 'False
	Tab(0).Control(3) = "cmdIf"
	Tab(0).Control(3).Enabled= 0 'False
E	Tab(0).Control(4) = "cmdLessThanOrEqualTo"
5	Tab(0).Control(4).Enabled= 0 'False
	Tab(0).Control(5) = "cmdGreaterThanEqualTo"
	Tab(0).Control(5).Enabled= 0 'False
	Tab(0).Control(6)= "cmdLessThan"
1.0	Tab(0).Control(6).Enabled= 0 'False
10	Tab(0).Control(7)= "cmdGreaterThan"
	Tab(0).Control(7).Enabled= 0 'False
	Tab(0).Control(8)= "cmdNotEqual"
	Tab(0).Control(8).Enabled= 0 'False
	Tab(0).Control(9)= "cmdAbs"
15	Tab(0).Control(9).Enabled= 0 'False
	Tab(0).Control(10)= "cmdFactorial"
	Tab(0).Control(10).Enabled= 0 'False
	Tab(0).Control(11)= "cmdExponent"
_ F 1	Tab(0).Control(11).Enabled= 0 'False
201 01 11 25	Tab(0).Control(12)= "cmdQuotient"
<u> </u>	Tab(0).Control(12).Enabled= 0 'False
un.	Tab(0).Control(13)= "cmdList"
.က <u>ရိ</u> ုဗ နိုင	Tab(0).Control(13).Enabled= 0 'False
	Tab(0).Control(14)= "cmdModulus"
25	Tab(0).Control(14).Enabled= 0 'False
	Tab(0).Control(15)= "cmdEqual"
e ex	Tab(0).Control(15).Enabled= 0 'False
	Tab(0).Control(16)= "cmdDivide"
%d 	Tab(0).Control(16).Enabled= 0 'False
3 6	Tab(0).Control(17)= "cmdMultiply"
-1	Tab(0).Control(17).Enabled= 0 'False
	Tab(0).Control(18)= "cmdMinus"
	Tab(0).Control(18).Enabled= 0 'False
	Tab(0).Control(19)= "cmdPlus"
35	Tab(0).Control(19).Enabled= 0 'False
	Tab(0).Control(20)= "cmdParens"
	Tab(0).Control(20).Enabled= 0 'False
	Tab(0).ControlCount= 21
	TabCaption(1) = "Variables"
40	TabPicture(1) = "Constraint.frx":001C
	Tab(1).ControlEnabled= 0 'False
	Tab(1).Control(0)= "cboVariableNames"
	Tab(1).Control(0).Enabled= 0 'False
	Tab(1).Control(1)= "cmdInsertVN"
45	Tab(1).Control(1).Enabled= 0 'False

```
Tab(1).ControlCount= 2
          TabCaption(2) = "Functions"
          TabPicture(2) = "Constraint.frx":0038
          Tab(2).ControlEnabled= 0 'False
 5
          Tab(2).Control(0)= "cboFunction"
          Tab(2).Control(0).Enabled= 0 'False
          Tab(2).Control(1)= "cmdInsertFunction"
          Tab(2).Control(1).Enabled= 0 'False
          Tab(2).Control(2)= "txtFunctionDescription"
10
          Tab(2).Control(2).Enabled= 0 'False
          Tab(2).ControlCount= 3
          Begin VB.CommandButton cmdParens
            Caption
                        = "()"
            BeginProperty Font
                         = "MS Sans Serif"
15
              Name
              Size
                        = 9.75
                         = 0
              Charset
                         = 400
              Weight
                          = 0 'False
              Underline
                       = 0 'False
              Italic
              Strikethrough = 0 'False
            EndProperty
            Height
                       = 375
            Left
                      = 2280
25
                        = 32
            TabIndex
            ToolTipText = "List"
                       = 1320
            Top
            Width
                       = 495
          End
          Begin VB.ComboBox cboFunction
            Height
                       = 315
            ItemData
                        = "Constraint.frx":0054
                      = -74400
            Left
                      = "Constraint.frx":007C
            List
                      = 2 'Dropdown List
35
            Style
            TabIndex
            ToolTipText = "Select a Prolog function from the list."
                       = 840
            Top
                       = 2175
            Width
40
          End
          Begin VB.CommandButton cmdInsertFunction
                       = "Insert"
            Caption
            Height
                       = 315
            Left
                      = -72120
```

45

TabIndex

= 30

```
= "Click here to insert this function into the constraint above at the current
            ToolTipText
       cursor position."
            Top
                       = 840
                        = 855
            Width
 5
           End
           Begin VB.TextBox txtFunctionDescription
            Height
                        = 1455
                       = -74400
            Left
            Locked
                         = -1 'True
                         = -1 'True
10
            MultiLine
                         = 2 'Vertical
            ScrollBars
            TabIndex
                         = 29
            ToolTipText = "The description of the function appears in this window."
            Top
                       = 1320
15
            Width
                        = 3135
           End
           Begin VB.ComboBox cboVariableNames
            Height
                        = 315
                         = "Constraint.frx":00EF
            ItemData
                       = -74400
            Left
                      = "Constraint.frx":0117
            List
 Œ٦
25.
            Style
                       = 2 'Dropdown List
            TabIndex
                         = 28
            ToolTipText = "Select a Prolog function from the list."
            Top
                       = 1320
 Į.
            Width
                        = 2175
           End
           Begin VB.CommandButton cmdInsertVN
                        = "Insert"
            Caption
            Height
                        = 315
            Left
                       = -72120
            TabIndex
                         = 27
            ToolTipText = "Click here to insert this variable name into the constraint above at the
       current cursor position."
35 '
            Top
                       = 1320
            Width
                        = 855
           End
           Begin VB.CommandButton cmdPlus
                        = "+"
            Caption
40
            BeginProperty Font
                          = "MS Sans Serif"
              Name
              Size
                        = 9.75
                          = 0
              Charset
              Weight
                          = 400
```

Underline

= 0 'False

```
Italic
                      = 0 'False
             Strikethrough = 0 'False
           EndProperty
           Height
                      = 375
 5
           Left
                     = 480
           TabIndex
                       = 25
           ToolTipText = "Plus"
                     = 840
           Top
           Width
                      = 495
10
          End
          Begin VB.CommandButton cmdMinus
           Caption
                      = "-"
           BeginProperty Font
             Name
                        = "MS Sans Serif"
15
             Size
                      = 9.75
             Charset
                        = 0
             Weight = 400
             Underline = 0 'False
                      = 0 'False
             Italic
Strikethrough = 0 'False
           EndProperty
           Height
                      = 375
                     = 1080
           Left
           TabIndex
                       = 24
25
           ToolTipText = "Minus"
 47
           Top
                     = 840
           Width
                      = 495
         End
          Begin VB.CommandButton cmdMultiply
                      = "*"
           Caption
           BeginProperty Font
             Name
                      = "MS Sans Serif"
             Size
                      = 9.75
                        = 0
             Charset
35
             Weight
                        = 400
                        = 0 'False
             Underline
                      = 0 'False
             Italic
             Strikethrough = 0 'False
           EndProperty
           Height
40
                      = 375
           Left
                     = 1680
                       = 23
           TabIndex
           ToolTipText = "Multiply"
                     = 840
           Top
           Width
                      = 495
45
```

```
End
          Begin VB.CommandButton cmdDivide
            Caption = "/"
            BeginProperty Font
 5
             Name
                        = "MS Sans Serif"
             Size
                       = 9.75
             Charset
                        = 0
                        = 400
             Weight
             Underline = 0 'False
10
             Italic
                      = 0 'False
             Strikethrough = 0 'False
            EndProperty
           Height
                      = 375
                     = 2280
            Left
15
                        = 22
            TabIndex
           ToolTipText = "Divide"
            Top
                      = 840
            Width
                      = 495
          End
          Begin VB.CommandButton cmdEqual
                       = "="

    Caption

 đì
            BeginProperty Font
 May after the
                        = "MS Sans Serif"
             Name
             Size
                       = 9.75
25=
             Charset
                       = 0
 1
             Weight
                         = 400
             Underline = 0 'False
             Italic
                    = 0 'False
             Strikethrough = 0 'False
            EndProperty
            Height
                      = 375
            Left
                     = 480
            TabIndex
                        = 21
            ToolTipText = "Equals"
35
            Top
                      = 1800
            Width
                      = 495
          End
          Begin VB.CommandButton cmdModulus
                       = "%"
            Caption
40
            BeginProperty Font
             Name
                        = "MS Sans Serif"
             Size
                       = 9.75
             Charset
                        = 0
             Weight
                        = 400
45
                         = 0 'False
             Underline
```

```
Italic
                      = 0 'False
             Strikethrough = 0 'False
            EndProperty
            Height
                      = 375
 5
            Left
                     = 2880
            TabIndex
                       = 20
            ToolTipText = "Modulo"
            Top
                      = 840
            Width
                      = 495
10
          End
          Begin VB.CommandButton cmdList
            Caption
                       = "([1,2])"
            BeginProperty Font
             Name
                        = "MS Sans Serif"
15
             Size
                       = 9.75
             Charset
                        = 0
                        = 400
             Weight
             Underline
                         = 0 'False
                      = 0 'False
             Italic
             Strikethrough = 0 'False
            EndProperty
 Ų,
            Height
                      = 375
 = 2880
            Left
            TabIndex
                        = 19
25
            ToolTipText = "List"
 ij,
            Top
                      = 1320
            Width
                      = 1095
          End
          Begin VB.CommandButton cmdQuotient
            Caption
                       = "\"
            BeginProperty Font
             Name
                        = "MS Sans Serif"
                       = 9.75
             Size
             Charset
                        = 0
             Weight = 400
35
             Underline
                         = 0 'False
             Italic
                       = 0 'False
             Strikethrough = 0 'False
            EndProperty
40
            Height
                      = 375
                     = 480
            Left
                       = 18
            TabIndex
            ToolTipText = "Quotient"
                      = 1320
            Top
```

Width

```
End
          Begin VB.CommandButton cmdExponent
                       = "^"
            Caption
           BeginProperty Font
                        = "MS Sans Serif"
 5
             Name
                       = 9.75
             Size
             Charset
                        = 0
                        = 400
             Weight
             Underline
                         = 0 'False
                      = 0 'False
10
             Italic
             Strikethrough = 0 'False
           EndProperty
           Height
                      = 375
           Left
                     = 3480
15
                        = 17
           TabIndex
           ToolTipText = "Exponent"
                      = 840
           Top
           Width
                      = 495
          End
          Begin VB.CommandButton cmdFactorial
                       = "!"
           Caption
           BeginProperty Font
                        = "MS Sans Serif"
             Name
 Ţ
             Size
                       = 9.75
25-
             Charset
                       = 0
 ij,
             Weight
                        = 400
             Underline
                         = 0 'False
M. C.
             Italic
                      = 0 'False
             Strikethrough = 0 'False
           EndProperty
           Height
                      = 375
           Left
                     = 1080
           TabIndex
                        = 16
           ToolTipText = "Factorial"
35
           Top
                      = 1320
           Width
                      = 495
          End
          Begin VB.CommandButton cmdAbs
                       = "||"
           Caption
40
           BeginProperty Font
                        = "MS Sans Serif"
             Name
             Size
                       = 9.75
                        = 0
             Charset
                        = 400
             Weight
                         = 0 'False
45
             Underline
```

```
Italic
                      = 0 'False
             Strikethrough = 0 'False
            EndProperty
           Height
                      = 375
5
           Left
                     = 1680
           TabIndex
                       = 15
           ToolTipText = "Absolute value"
           Top
                      = 1320
            Width
                      = 495
10
          End
          Begin VB.CommandButton cmdNotEqual
            Caption
                       = "=/="
            BeginProperty Font
             Name
                        = "MS Sans Serif"
15
             Size
                       = 9.75
             Charset
                        = 0
                         = 400
             Weight
             Underline
                         = 0 'False
                      = 0 'False
             Italic
2<u>0</u>
             Strikethrough = 0 'False
Ð١
            EndProperty
25
           Height
                      = 375
                     = 1080
           Left
                        = 14
            TabIndex
            ToolTipText = "Does not equal"
            Top
                      = 1800
            Width
                       = 495
          End
          Begin VB.CommandButton cmdGreaterThan
                       = ">"
           Caption
           BeginProperty Font
             Name
                        = "MS Sans Serif"
             Size
                       = 9.75
                        = 0
             Charset
35
             Weight
                        = 400
             Underline
                         = 0 'False
                       = 0 'False
             Italic
             Strikethrough = 0 'False
            EndProperty
           Height
40
                       = 375
           Left
                     = 1680
                        = 13
            TabIndex
            ToolTipText = "Greater than"
            Top
                      = 1800
```

= 495

Width

```
End
          Begin VB.CommandButton cmdLessThan
                       = "<"
            Caption
            BeginProperty Font
                         = "MS Sans Serif"
 5
             Name
             Size
                       = 9.75
             Charset
                        = 0
                       \cdot = 400
             Weight
             Underline = 0 'False
                       = 0 'False
10
             Italic
             Strikethrough = 0 'False
            EndProperty
           Height
                       = 375
           Left
                     = 2280
15
                        = 12
            TabIndex
            ToolTipText = "Less than"
            Top
                      = 1800
            Width
                       = 495
          End
          Begin VB.CommandButton cmdGreaterThanEqualTo
                       = ">="
            Caption
           BeginProperty Font
                        = "MS Sans Serif"
             Name
             Size
                       = 9.75
25.
             Charset
                       = 0
 1
             Weight
                         = 400
             Underline
                         = 0 'False
             Italic
                       = 0 'False
             Strikethrough = 0 'False
            EndProperty
           Height
                       = 375
           Left
                     = 2880
           TabIndex
                        = 11
            ToolTipText = "Greater than or equal to"
35
            Top
                      = 1800
                       = 495
            Width
          End
          Begin VB.CommandButton cmdLessThanOrEqualTo
                       = "<="
            Caption
40
            BeginProperty Font
                         = "MS Sans Serif"
             Name
             Size
                       = 9.75
                        = 0
             Charset
             Weight
                         = 400
45
             Underline
                         = 0 'False
```

```
Italic
                       = 0 'False
             Strikethrough = 0 'False
            EndProperty
           Height
                       = 375
 5
                     = 3480
           Left
            TabIndex
                       = 10
            ToolTipText = "Less than or equal to"
                      = 1800
            Top
            Width
                       = 495
10
          End
          Begin VB.CommandButton cmdIf
            Caption
                       = "if"
            BeginProperty Font
             Name
                        = "MS Sans Serif"
15
             Size
                       = 9.75
             Charset ·
                        = 0
                        = 400
             Weight
             Underline
                         = 0 'False
                      = 0 'False
             Italic
             Strikethrough = 0 'False
201
 (I)
            EndProperty
25 m
           Height
                       = 375
           Left
                     = 480
           TabIndex
                        = 9
            ToolTipText = "If"
            Top
                      = 2280
            Width
                       = 735
          End
          Begin VB.CommandButton cmdThen
                       = "then"
            Caption
            BeginProperty Font
             Name
                        = "MS Sans Serif"
                       = 9.75
             Size
                        = 0
             Charset
35
             Weight
                        = 400
             Underline
                         = 0 'False
             Italic
                       = 0 'False
             Strikethrough = 0 'False
            EndProperty
           Height
40
                       = 375
           Left
                     = 1320
                        = 8
            TabIndex
            ToolTipText = "then"
            Top
                      = 2280
```

Width

```
End
          Begin VB.CommandButton cmdElse
                       = "else"
            Caption
            BeginProperty Font
             Name
                        = "MS Sans Serif"
 5
             Size
                       = 9.75
             Charset
                       = 0
                        = 400
             Weight
             Underline = 0 'False
10
                      = 0 'False
             Italic
             Strikethrough = 0 'False
            EndProperty
            Height
                      = 375
                     = 2160
           Left
15
                        = 7
            TabIndex
            ToolTipText = "else"
                      = 2280
            Top
            Width
                      = 735
          End
201
          Begin VB.CommandButton cmdElseIf
 ġ1
           Caption
                       = "elseif"
           BeginProperty Font
                        = "MS Sans Serif"
             Name
             Size
                       = 9.75
             Charset
                       = 0
             Weight
                        = 400
             Underline = 0 'False
Italic
                      = 0 'False
             Strikethrough = 0 'False
3₽.
            EndProperty
           Height
                      = 375
            Left
                     = 3000
            TabIndex
                        = 6
            ToolTipText = "elseif"
35
            Top
                      = 2280
                      = 975
            Width
          End
        Begin VB.TextBox txtConstraint
40
          Height
                     = 315
          Left
                    = 240
          TabIndex
                      = 3
          ToolTipText = "Enter the constraint here."
          Top
                    = 480
          Width
                     = 4455
45
```

```
End
         Begin VB.TextBox txtComment
                     = 1335
          Height
                    = 240
          Left
                       = -1 'True
 5
          MultiLine
          TabIndex
                       = 0
                     = 4800
          Top
          Width
                     = 4455
         End
10
         Begin VB.CommandButton cmdConOK
                      = "OK"
          Caption
          Default
                     = -1 'True
                     = 495
          Height
          Left
                    = 4920
15
          TabIndex
                       = 1
          ToolTipText = "Click here to save this constraint."
                     = 120
          Top
          Width
                     = 1215
         End
         Begin VB.CommandButton cmdConCancel
203
                      = "Cancel"
          Caption
Height
                     = 495
          Left
                    = 4920
          TabIndex
                       = 2
          ToolTipText = "Click here to return without creating or-modifying this constraint."
         Top
                     = 720
          Width
                     = 1215
The state of the
         End
         Begin VB.Label lblComment
3Q:
          Caption
                      = "Comment"
Height
                     = 255
          Left
                    = 240
          TabIndex
                       = 26
          Top
                     = 4560
35
          Width
                     = 1215
         End
         Begin VB.Label lblConstraints
          Caption
                      = "Constraint"
                     = 255
          Height
40
          Left
                    = 240
          TabIndex
          ToolTipText = "Click on the down arrow for function prototypes"
                     = 240
          Top
                     = 1695
          Width
45
         End
```

```
Attribute VB Name = "frmConstraints"
       Attribute VB GlobalNameSpace = False
       Attribute VB Creatable = False
       Attribute VB PredeclaredId = True
 5
       Attribute VB_Exposed = False
       Option Explicit
       Private mbytAddEditFlag As Byte
       Private mlstListBox As ListBox
       Private mudtCon As Constraint
10
       Private mudtModel As Model
       Private mudtConType As ConstraintType
       Private Enum ResourceStrings
         rcStartFunctions = 101
rcEndFunctions = 125
         rcStartExplanations = 201
       End Enum
       Private mblnChangeFocus As Boolean
       Public Property Let AddEditFlag(ByVal bytNewValue As Byte)
         mbytAddEditFlag = bytNewValue
       End Property
       Public Property Let ListBox(ByVal lstNewValue As ListBox)
         Set mlstListBox = lstNewValue
       End Property
       Public Property Let Constraint(ByVal udtNewValue As Constraint)
25
         Set mudtCon = udtNewValue
       End Property
       Public Property Let ConstraintType(ByVal udtNewValue As ConstraintType)
```

VBSCA -42-

End

```
mudtConType = udtNewValue
        End Property
        Public Property Let Model(ByVal udtNewValue As Model)
          Set mudtModel = udtNewValue
 5
        End Property
        Private Sub cboFunction Click()
          Dim intl As Integer
          For intI = 0 To cboFunction.ListCount - 1
            If cboFunction = cboFunction.List(intI) Then
10
              txtFunctionDescription = LoadResString(intI + rcStartExplanations)
               Exit For
            End If
          Next intI
          If mblnChangeFocus Then
            txtConstraint.SetFocus
          End If
201
        End Sub
       Private Sub cboVariableNames_Click()
          If mblnChangeFocus Then
            txtConstraint.SetFocus
          End If
        End Sub
       Private Sub cmdElse Click()
          Call InsertText("else", 0)
30
       End Sub
       Private Sub cmdElseIf_Click()
          Call InsertText("elseif", 0)
```

```
End Sub
        Private Sub cmdGreaterThan Click()
          Call InsertText(">", 0)
 5
        End Sub
       Private Sub cmdGreaterThanEqualTo_Click()
          Call InsertText(">=", 0)
        End Sub
       Private Sub cmdIf_Click()
10
          Call InsertText("if", 0)
       End Sub
       Private Sub cmdParens Click()
          Call InsertText("()", 1)
       End Sub
       Private Sub cmdThen_Click()
 C)
          Call InsertText("then", 0)
       End Sub
       Private Sub cmdInsertFunction Click()
          If cboFunction = "brandom()" Or cboFunction = "random()" Then
            Call InsertText(cboFunction, 0)
25
          Else
            Call InsertText(cboFunction, 1)
          End If
       End Sub
       Private Sub cmdInsertVN_Click()
30
          Call InsertText(cboVariableNames, 0)
```

```
End Sub
       Private Sub cmdLessThan_Click()
          Call InsertText("<")</pre>
 5
        End Sub
       Private Sub cmdLessThanOrEqualTo_Click()
          Call InsertText("<=", 0)
10
        End Sub
       Private Sub cmdNotEqual_Click()
          Call InsertText("=/=", 0)
End Sub
       Private Sub cmdPlus Click()
          Call InsertText("+")
       End Sub
       Private Sub cmdMinus_Click()
          Call InsertText("-")
       End Sub
       Private Sub cmdMultiply Click()
          Call InsertText("*")
       End Sub
       Private Sub cmdDivide_Click()
          Call InsertText("/")
25
       End Sub
```

```
Private Sub cmdModulus Click()
          Call InsertText("%")
        End Sub
       Private Sub cmdEqual_Click()
          Call InsertText("=")
        End Sub
        Private Sub cmdList_Click()
          Call InsertText("([])", 2)
        End Sub
       Private Sub cmdQuotient_Click()
          Call InsertText("\")
        End Sub
       Private Sub cmdExponent_Click()
          Call InsertText("^")
        End Sub
       Private Sub cmdFactorial_Click()
          Call InsertText("!")
        End Sub
       Private Sub cmdAbs_Click()
          Call InsertText("||", 1)
20
       End Sub
       Private Sub InsertText(ByVal strInsertedText As String, _
          Optional ByVal intOffset As Integer = -1)
```

	Dim strFront As String Dim strBack As String
	If intOffset = -1 Then intOffset = Len(strInsertedText) - 1
5	strFront = left(txtConstraint, txtConstraint.SelStart) strBack = right(txtConstraint, Len(txtConstraint) txtConstraint.SelStart - txtConstraint.SelLength)
10	txtConstraint = strFront & strInsertedText & strBack txtConstraint.SetFocus
	' move the cursor txtConstraint.SelStart = Len(strFront) + Len(strInsertedText) - intOffset
15	End Sub
	Private Sub Command3_Click()
	End Sub
Man Carlo	Private Sub Form_Load()
1	' disable OK button if changes aren't allowed If mudtModel.IsFrozen Then cmdConOK.Enabled = False
	Else cmdConOK.Enabled = True End If
23 E1	Dim udtV As Variable
	' load variable names into combo box cboVariableNames.Clear
30	For Each udtV In mudtModel.Variables Call cboVariableNames.AddItem(udtV.name) Next udtV
35	If mbytAddEditFlag = aeEdit Then txtConstraint = mudtCon.ConstraintString txtComment = mudtCon.Comment End If
40	'load functions into combo box Dim intI As Integer

```
For intI = rcStartFunctions To rcEndFunctions
            cboFunction.List(intI - rcStartFunctions) = LoadResString(intI)
          Next intI
 5
          mblnChangeFocus = False
          If cboVariableNames.ListCount > 0 Then
            cboVariableNames.ListIndex = 0
          End If
10
          cboFunction.ListIndex = 0
          mblnChangeFocus = True
       End Sub
       Private Sub cmdConOK Click()
          If Len(txtConstraint) = 0 Then
15
            Call MsgBox("Null constraints are not permitted", vbExclamation, "Error")
            Exit Sub
          End If
 ű
          If mbytAddEditFlag = aeEdit Then ' we're editing an old one
            'update the constraint with new data from the form
            Call mudtCon.Update(txtConstraint, mudtConType, txtComment)
            ' update the text in the list box
            mlstListBox.List(mlstListBox.ListIndex) = mudtCon.ConstraintString
          Else
            ' Add the new constraint
            Set mudtCon = mudtModel.Constraints.Add(txtConstraint, True, _
               mudtConType, txtComment)
            With mlstListBox
               ' Add the new constraint to the list box
               Call .AddItem(mudtCon.ConstraintString)
               'Set ItemData to index value of the variable object
30
               .ItemData(.ListCount - 1) = mudtCon.index
               'Check the check box
               .Selected(.ListCount - 1) = True
            End With
          End If
35
          Call frmTCA.AddUndefinedVariables(txtConstraint)
          Unload Me
40
       End Sub
```

Private Sub cmdConCancel_Click()

Unload Me

5 End Sub

```
'EditConstraint.frm
       VERSION 5.00
       Begin VB.Form frmEditText
         BorderStyle = 1 'Fixed Single
 5
         ClientHeight = 1455
         ClientLeft
                     = 45
         ClientTop
                     = 330
         ClientWidth = 4785
         LinkTopic
                     = "Form1"
10
         MaxButton
                      = 0 'False
                      = 0 'False
         MinButton
         ScaleHeight
                      = 1455
                      = 4785
         ScaleWidth
         StartUpPosition = 3 'Windows Default
15
         Begin VB.CommandButton cmdEditTextOK
                      = "OK"
          Caption
          Default
                     = -1 'True
                     = 495
          Height
                    = 3360
          Left
= .2
          TabIndex
          Top
                     = ,120
          Width
                     = 1215
         End
         Begin VB.CommandButton cmdEditTextnCancel
2≨‡
                      = "Cancel"
          Caption
          Height
                     = 495
30.
          Left
                    = 3360
          TabIndex
                       = 1
          Top
                     = 720
          Width
                     = 1215
         End
         Begin VB.TextBox txtEditText
          Alignment
                       = 2 'Center
          Height
                     = 375
35
          Left
                    = 240
          TabIndex
                      = 0
          Top
                     = 120
                     = 2895
          Width
         End
40
       End
       Attribute VB Name = "frmEditText"
       Attribute VB GlobalNameSpace = False
       Attribute VB Creatable = False
       Attribute VB PredeclaredId = True
```

Attribute VB_Exposed = False
Option Explicit
'These are used as references to the ListBox in frmTCA currently being editted
Public lstListBox As ListBox
Public intInd As Integer

Private Sub cmdEditTextnCancel_Click()
Unload Me
End Sub

```
'Form1.frm
       VERSION 5.00
       Begin VB.Form Form1
                    = "Form1"
        Caption
 5
        ClientHeight = 4050
        ClientLeft
                    = 60
        ClientTop
                     = 345
        ClientWidth = 5595
        LinkTopic
                     = "Form1"
        ScaleHeight
                      = 4050
10
                      = 5595
        ScaleWidth
        StartUpPosition = 3 'Windows Default
        Begin VB.CommandButton Command1
                      = "Clear"
          Caption
15
          Height
                     = 1455
          Left
                    = 3720
          TabIndex
                       = 2
          Top
                     = 2520
          Width
                     = 1455
End
        Begin VB.TextBox Text1
          Height
                     = 855
          Left
                    = 600
          TabIndex
                      = 1
                    = "Text1"
          Text
 HI WE WILL HE
                    = 960
          Top
          Width
                     = 2175
        End
        Begin VB.CommandButton cmdRun
                      = "Run"
          Caption
                     = 1335
          Height
          Left
                    = 3720
          TabIndex
                       = 0
          Top
                     = 960
35
          Width
                     = 1455
        End
       End
       Attribute VB Name = "Form1"
       Attribute VB GlobalNameSpace = False
40
       Attribute VB Creatable = False
       Attribute VB PredeclaredId = True
       Attribute VB Exposed = False
       Option Explicit
```

```
Private Sub cmdRun Click()
         Dim udtP As New Prolog
         Dim lngR As Long
         If udtP.StartProlog("hlp4lib.p4") = False Then
 5
            Call MsgBox("Prolog failure on startup", vbExclamation, "Error")
          End If
         Call udtP.AddVariable("int(I),[520<=I<=590 step 5], int(I2),[I + 5<=I2<=I + 30 step 1]")
10
         lngR = udtP.SolveConstraintsOrdered(1)
          Text1 = Str(lngR)
       End Sub
       Private Sub Command1_Click()
15
          Text1 = ""
 ij
 End Sub
```

```
' frmAbout.frm
       VERSION 5.00
       Begin VB.Form frmAbout
         BorderStyle
                      = 4 'Fixed ToolWindow
 5
         Caption
                    = "About TCA"
         ClientHeight = 2610
         ClientLeft
                     = 45
         ClientTop
                     = 285
         ClientWidth = 4440
                      = "Form1"
10
         LinkTopic
         LockControls = -1 'True
                      = 0 'False
         MaxButton
                      = 0 \cdot 'False
         MinButton
         ScaleHeight
                      = 2610
15
                      = 4440
         ScaleWidth
         ShowInTaskbar = 0 'False
         StartUpPosition = 1 'CenterOwner
         Begin VB.CommandButton cmdOK
          Caption
                      = "OK"
          Height
                      = 495
The Best of the Best of
          Left
                    = 3120
          TabIndex
                       = 1
          Top
                     = 120
          Width
                      = 1215
25
         End
         Begin VB.Label lblVersion
 ILL HER ILL
          Height
                      = 255
          Left
                     = 240
          TabIndex
                       = 2
          Top
                     = 2160
          Width
                      = 2295
         End
         Begin VB.Label Label1
                      = "TCA is a collaborative development of the Assessment and Research
          Caption
35
       Divisions."
          Height
                      = 615
          Left
                    = 240
          TabIndex
                       = 0
          Top
                     = 1320
40
          Width
                      = 2535
         End
         Begin VB.Image imaETS
          BorderStyle = 1 'Fixed Single
                      = 780
          Height
```

```
Left
                      = 960
           Picture
                       = "frmAbout.frx":0000
           Top
                      = 240
           Width
                       = 1275
 5
         End
        End
        Attribute VB Name = "frmAbout"
        Attribute VB GlobalNameSpace = False
        Attribute VB Creatable = False
        Attribute VB PredeclaredId = True
10
        Attribute VB Exposed = False
        Option Explicit
        Private Sub cmdEasterEgg MouseDown(Button As Integer, Shift As Integer, X As Single, Y As
        Single)
          If Button = vbRightButton Then
15
            ' display easter egg
            Beep
 C)
          End If
 43
 <u>D</u>1
 Ū
        End Sub
201
        Private Sub cmdOK_Click()
          Unload Me
        End Sub
        Private Sub Form_Load()
          lblVersion = frmSplash.lblVersion
        End Sub
       Private Sub imaETS DblClick()
          ' display easter egg
          Beep
30
       End Sub
        ' frmAttributes.frm
        VERSION 5.00
```

```
Begin VB.Form frmAttributes
        BorderStyle = 4 'Fixed ToolWindow
                    = "Family Attributes"
        Caption
        ClientHeight = 1590
 5
        ClientLeft
                    = 45
        ClientTop
                     = 285
        ClientWidth = 4305
        LinkTopic
                     = "Form1"
        LockControls = -1 'True
                      = 0 'False
10
        MaxButton
                      = 0 'False
        MinButton
        ScaleHeight = 1590
        ScaleWidth = 4305
        ShowInTaskbar = 0 'False
        StartUpPosition = 1 'CenterOwner
15
        Begin VB.ComboBox cboProximity
          Height
                     = 315
                      = "frmAttributes.frx":0000
          ItemData
          Left
                    = 240
                    = "frmAttributes.frx":000D
          List
                    = 2 'Dropdown List
          Style
 May May Kan Kah
          TabIndex
                      = 4
                    = 360
          Top
          Width
                     = 1935
25
        End
 4
        Begin VB.OptionButton optGeneric
                      = "Generic"
          Caption
          Height
                     = 195
          Index
                     = 0
          Left
                    = 120
          TabIndex
                      = 3
                    = 1035
          Top
          Value
                     = -1 'True
          Width
                     = 975
35
        End
        Begin VB.OptionButton optGeneric
                     = "Non-generic"
          Caption
          Height
                     = 195
                     = 1
          Index
40
          Left
                    = 1080
          TabIndex
                      = 2
          Top
                    = 1035
          Width
                     = 1455
        End
```

Begin VB.CommandButton cmdCancel

```
= "Cancel"
          Caption
          Height
                      = 495
          Left
                     = 3000
          TabIndex
                       = 1
                        = "Click here to return without saving these family attributes."
 5
          ToolTipText
                     = 720
          Top
           Width
                      = 1215
         End
         Begin VB.CommandButton cmdOK
                     · = "OK"
10
           Caption
          Default
                      = -1 'True
          Height
                      = 495
          Left
                     = 3000
          TabIndex
                       = 0
                        = "Click here to save these family attributes."
           ToolTipText
15
           Top
                     = 120
           Width
                      = 1215
         End
         Begin VB.Label lbl
= "Variant proximity"
           Caption
           Height
                      = 255
           Left
                     = 240
                       = 5
           TabIndex
                     = 120
           Top
           Width
                      = 1335
         End
       End
       Attribute VB Name = "frmAttributes"
       Attribute VB GlobalNameSpace = False
       Attribute VB Creatable = False
       Attribute VB PredeclaredId = True
       Attribute VB Exposed = False
       Option Explicit
       Private mblnOK As Boolean
       Private mblnGeneric As Boolean
35
       Private mudtProximity As Proximity
       Private Sub Form Load()
          mblnOK = False
          cboProximity.ListIndex = frmTCA.Family.Proximity
40
          If frmTCA.Family.Generic Then
```

```
optGeneric(0) = True
         Else
            optGeneric(1) = True
         End If
 5
         mblnGeneric = frmTCA.Family.Generic
         mudtProximity = frmTCA.Family.Proximity
       End Sub
       Public Property Get Proximity() As Proximity
         Proximity = mudtProximity
10
       End Property
       Public Property Get Generic() As Boolean
         Generic = mblnGeneric
       End Property
       Private Sub cmdOK Click()
         mblnOK = True
         Unload Me
       End Sub
       Private Sub cmdCancel_Click()
         Unload Me
       End Sub
       Public Property Get OK() As Boolean
         OK = mblnOK
25
       End Property
       Private Sub cboProximity Click()
         mudtProximity = cboProximity.ListIndex
```

End Sub

Private Sub optGeneric_Click(Index As Integer)

mblnGeneric = optGeneric(0)

End Sub

```
' frmComments.frm
       VERSION 5.00
       Begin VB.Form frmComments
        BorderStyle = 4 'Fixed ToolWindow
 5
        Caption
                   = "Comments"
        ClientHeight = 3765
                   = 45
        ClientLeft
        ClientTop
                    = 285
        ClientWidth = 5250
10
        LinkTopic
                     = "Form1"
        LockControls = -1 'True
                     = 0 'False
        MaxButton
        MinButton
                     = 0 'False
        ScaleHeight = 3765
15
                     = 5250
        ScaleWidth
        ShowInTaskbar = 0 'False
        StartUpPosition = 2 'CenterScreen
        Begin VB.CommandButton cmdCancel
                     = "Cancel"
          Caption
          Height
                     = 495
          Left
                   = 3960
25
          TabIndex
                      = 2
          ToolTipText = "Click here to save these family attributes."
          Top
                    = 720
                     = 1215
          Width
        End
        Begin VB.CommandButton cmdOK
                     = "OK"
          Caption
          Default
                     = -1 'True
          Height
                     = 495
          Left
                    = 3960
          TabIndex
          ToolTipText = "Click here to save these family attributes."
          Top
                    = 120
          Width
                     = 1215
35
        End
        Begin VB.TextBox txtComment
          Height
                     = 3495
          Left
                    = 120
40
          MultiLine
                      = -1 'True
                      = 0
          TabIndex
                    = 120
          Top
          Width
                     = 3735
        End
```

```
End
        Attribute VB Name = "frmComments"
        Attribute VB GlobalNameSpace = False
        Attribute VB Creatable = False
       Attribute VB PredeclaredId = True
 5
       Attribute VB Exposed = False
       Private mstrComment As String
       Public Property Get Comment() As String
          Comment = mstrComment
10
        End Property
       Public Property Let Comment(ByVal strNewValue As String)
          txtComment = strNewValue
          mstrComment = strNewValue
        End Property
 to 15th Ann Bill Ann Com Com
       Private Sub cmdCancel Click()
          Unload Me
       End Sub
201
       Private Sub cmdOK_Click()
          mstrComment = txtComment
          Unload Me
```

C)

End Sub

```
' frmDifficulty.frm
       VERSION 5.00
       Object = "{6B7E6392-850A-101B-AFC0-4210102A8DA7}#1.3#0"; "COMCTL32.OCX"
       Begin VB.Form frmDifficulty
                    = 4 'Fixed ToolWindow
 5
         BorderStyle
         ClientHeight = 8730
         ClientLeft
                    = 45
         ClientTop
                     = 285
         ClientWidth = 6855
10
         LinkTopic
                     = "Form1"
         LockControls = -1 'True
                      = 0 'False
         MaxButton
         MinButton
                      = 0 'False
         ScaleHeight
                     = 8730
15
         ScaleWidth
                      = 6855
         ShowInTaskbar = 0 'False
         StartUpPosition = 2 'CenterScreen
         Begin VB.CheckBox chkRoute
                      = "Route to TCS"
          Caption
          Height
                     = 375
 Man Han Man Han
          Left
                    = 2640
                       = 33
          TabIndex
          Top
                     = 1800
          Width
                     = 1935
25
         End
         Begin VB.ComboBox cboKey
          Height
                     = 315
                      = "frmDifficulty.frx":0000
          ItemData
          Left
                    = 2640
          List
                    = "frmDifficulty.frx":0013
          Style
                    = 2 'Dropdown List
                       = 30
          TabIndex
          Top
                     = 1200
          Width
                     = 615
35
         End
         Begin VB.CheckBox chkCalcDifficulty
                      = "Calculate difficulty"
          Caption
          Height
                     = 255
          Left
                    = 240
40
          TabIndex
                      = 27
          Top
                     = 3600
                     = 1 'Checked
          Value
          Width
                     = 1935
         End
```

```
Begin VB.ComboBox cboDeliveryMode
                     = 315
          Height
          ItemData
                      = "frmDifficulty.frx":0026
          Left
                    = 2640
                   = "frmDifficulty.frx":0030
 5
          List
          Style
                    = 2 'Dropdown List
          TabIndex
                      = 25
                    = 480
          Top
          Width
                     = 1695
10
        End
        Begin VB.ComboBox cboDomain
          Height
                     = 315
          ItemData
                      = "frmDifficulty.frx":003E
          Left
                   = 240
15
          List
                   = "frmDifficulty.frx":004E
          Style
                    = 2 'Dropdown List
                      = 18
          TabIndex
                    = 1200
          Top
          Width
                     = 1695
        End
        Begin VB.OptionButton optNature
= "Pure"
          Caption
          Height
                     = 375
          Index
                    = 0
          Left
                    = 240
          TabIndex
                      = 17
                    = 1800
          Top
          Value
                     = -1 'True
          Width
                     = 735
        Begin VB.OptionButton optNature
          Caption
                     = "Real"
          Height
                     = 375
          Index
                     = 1
          Left
                    = 1200
35
          TabIndex
                      = 16
          Top
                    = 1800
          Width
                     = 735
        End
40
        Begin VB.CommandButton cmdOK
          Caption
                     = "OK"
          Default
                     = -1 'True
          Height
                     = 495
          Left
                    = 5520
```

TabIndex

```
ToolTipText = "Click here to save changes and return."
                    = 240
          Top
          Width
                     = 1215
        End
        Begin VB.CommandButton cmdCancel
 5
          Caption
                     = "Cancel"
          Height
                     = 495
          Left
                    = 5520
          TabIndex
                      = 7
          ToolTipText = "Click here to save changes and return."
10
          Top
                    = 840
          Width
                     = 1215
        End
        Begin VB.TextBox txtBatchId
15
          Height
                     = 315
          Left
                    = 240
          TabIndex
                      = 0
          Top
                    = 480
          Width
                     = 1695
ري
رو2
        End
 16. Jan 6. 6.1
        Begin ComctlLib.Slider sldTDEstimate
          Height
                     = 375
          Left
                    = 480
          TabIndex
                     = 20
          Top
                    = 2760
          Width
                     = 3975
          ExtentX
                     = 7011
          ExtentY
                      = 661
                      = 327682
          Version
          LargeChange = 1
          Min
                    = 1
          Max
                     = 5
          SelStart
                     = 1
          Value
                     = 1
35
        End
        Begin VB.Frame fraPredDiff
          Caption
                     = "Predicted Difficulty"
          Height
                     = 1575
          Left
                    = 480
40
          TabIndex
                      = 10
                    = 6720
          Top
          Width
                     = 4575
          Begin ComctlLib.Slider sldDiffEstimate
                      = 375
           Height
```

Left

```
TabIndex
                       = 11
                     = 720
           Top
           Width
                      = 3975
            ExtentX
                       = 7011
           _ExtentY
5
                       = 661
            Version
                       = 327682
           Min
                      = 1
                      = 5
           Max
           SelStart
                      = 1
10
           Value
                      = 1
          End
          Begin VB.Label lblIRTValue
           Height
                      = 255
                     = 1080
           Left
15
           TabIndex
                       = 32
           Top
                     = 360
           Width
                      = 3015
          End
          Begin VB.Label lblPredEasy
                      = "Easy"
20
           Caption
ð١
           Height
                      = 255
25
           Left
                     = 3840
           TabIndex
                       = 15
                     = 1200
           Top
           Width
                      = 615
          End
30.
          Begin VB.Label lblPredMed
           Caption
                       = "Medium"
           Height
                      = 255
           Left
                     = 1920
TabIndex
                       = 14
                      = 1200
           Top
           Width
                      = 855
          End
35
          Begin VB.Label lblPredDiff
                       = "Difficult"
           Caption
           Height
                      = 255
           Left
                     = 240
           TabIndex
                       = 13
40
                     = 1200
           Top
           Width
                      = 735
          End
          Begin VB.Label lblIRT
           Caption
                       = "IRT b:"
```

Height

= 255

```
Left
                     = 360
                        = 12
            TabIndex
                      = 360
            Top
                       = 495
            Width
5
          End
        End
        Begin VB.Frame fraGREDiff
                     = "GRE Difficulty"
          Caption
          Height
                     = 4575
          Left
10
                    = 240
          TabIndex
                      = 2
                     = 3960
          Top
          Width
                     = 5055
          Begin VB.ComboBox cboGREConcept
15
            Height
                       = 315
                        = "frmDifficulty.frx":0080
            ItemData
            Left
                     = 240
                     = "frmDifficulty.frx":0093
            List
            Style
                      = 2 'Dropdown List
201
            TabIndex
                        = 28
 (Ti
                      = 2160
            Top
25
            Width
                       = 2055
          End
          Begin VB.ComboBox cboGRECog
            Height
                       = 315
            ItemData
                        = "frmDifficulty.frx":00ED
20
20
30
            Left
                     = 240
            List
                     = "frmDifficulty.frx":00FA
            Style
                      = 2 'Dropdown List
            TabIndex
                        = 5
                      = 1440
            Top
            Width
                       = 2055
          End
          Begin VB.ComboBox cboGREComp
35
            Height
                       = 315
                        = "frmDifficulty.frx":012D
            ItemData
            Left
                      = 240
                     = "frmDifficulty.frx":013D
            List
            Style
                      = 2 'Dropdown List
            TabIndex
                        = 3
40
            Top
                      = 720
                       = 2055
            Width
          End
          Begin VB.Label lblConcept
```

Caption

= "Concept:"

```
Height
                      = 255
            Left
                     = 240
            TabIndex
                        = 29
                      = 1920
            Top
 5
            Width
                      = 975
          End
          Begin VB.Label lblGRECog
                       = "Cognition:"
            Caption
           Height
                      = 255
10
           Left
                     = 240
           TabIndex
                        = 6
            Top
                      = 1200
            Width
                      = 975
          End
15
          Begin VB.Label lblGREComp
                       = "Computation:"
            Caption
           Height
                      = 255
           Left
                     = 240
            TabIndex
                        = 4
= 480
           Top
            Width
                      = 975
          End
        End
        Begin VB.Frame fraGMATDiff
          Caption
                     = "GMAT Difficulty"
          Height
                     = 4575
30. 0.0
          Left
                    = 240
          TabIndex
                      = 9
                  = 3960
          Top
          Width
                     = 5055
        End
        Begin VB.Frame fraOther
                     = 4575
          Height
          Left
                    = 240
35
          TabIndex
                      = 34
          Top
                    = 3960
          Width
                     = 5055
        End
        Begin VB.Label lblKey
40
          Caption
                     = "Key:"
          Height
                     = 255
          Left
                    = 2640
          TabIndex
                      = 31
          Top
                    = 960
                     = 975
45
          Width
```

```
End
         Begin VB.Label lblTarget
                     = "Target template:"
          Caption
          Height
                     = 255
          Left
                    = 2640
 5
          TabIndex
                     = 26
                    = 240
          Top
          Width
                     = 1815
         End
10
         Begin VB.Label lblSlideDirections
                      = "Adjust the slide to estimated variant difficulty:"
          Caption
          Height
                     = 255
          Left
                    = 600
          TabIndex
                      = 24
15
          Top
                    = 2400
          Width
                     = 3615
         End
         Begin VB.Label lblTDDiff
          Caption
                     = "Difficult"
20
          Height
                     = 255
          Left
                    = 480
 Ō١
May Am Com
          TabIndex
                      = 23
          Top
                    = 3240
          Width
                     = 735
         End
        Begin VB.Label lblTDMed
          Caption
                     = "Medium"
          Height
                     = 255
          Left
                    = 2160
          TabIndex
                      = 22
          Top
                    = 3240
          Width
                     = 855
        Begin VB.Label lblTDEasy
                     = "Easy"
35
          Caption
          Height
                     = 255
          Left
                    = 4080
          TabIndex
                      = 21
          Top
                    = 3240
40
          Width
                     = 615
        End
        Begin VB.Label lblDomain
                     = "Domain:"
          Caption
                     = 255
          Height
```

Left

= 240

```
TabIndex
                        = 19
                      = 960
          Top
           Width
                      = 975
         End
         Begin VB.Label LblBatch
 5
           Caption
                       = "Batch id:"
          Height
                      = 255
          Left
                     = 240
          TabIndex
                        = 1
10
          Top
                     = 240
                      = 975
           Width
         End
       End
       Attribute VB_Name = "frmDifficulty"
       Attribute VB GlobalNameSpace = False
15
       Attribute VB Creatable = False
       Attribute VB PredeclaredId = True
       Attribute VB Exposed = False
       Option Explicit
 ui]
20
       Dim mudtFamily As Family
       Dim mudtClone As Clone
 UT
 Mr. Man
       Dim mudtDE As DifficultyEstimate
       Dim mudtGreDE As GREDifficultyEstimate
 Mile April
       Dim mudtGmatDE As GMATDifficultyEstimate
       Dim mblnFormLoad As Boolean
 47
       Public Property Let Family(ByVal udtNewValue As Family)
 <u>ļ-</u>h
         Set mudtFamily = udtNewValue
       End Property
       Public Property Let Clone(ByVal udtNewValue As Clone)
30
         Set mudtClone = udtNewValue
       End Property
       Private Sub Form Load()
         Set mudtDE = mudtClone.DiffEst
         mblnFormLoad = True
35
```

```
' if there's a key, prohibit input.
                              If mudtFamily.ItemType = ptStandardMC Then
                                     cboKey.Enabled = False
                              Else
   5
                                     cboKey.Enabled = True
                              End If
                              'change form depending on program
                              Select Case mudtFamily.Program
10
                                     Case prGRE
                                            fraGREDiff.ZOrder
                                            fraPredDiff.ZOrder
                                     Case prGMAT
                                            fraGMATDiff.ZOrder
15
                                            fraPredDiff.ZOrder
                                     Case Else
                                            fraOther.ZOrder
                              End Select
200
                              cboDomain.ListIndex = mudtClone.Domain
                              txtBatchId = mudtClone.BatchID
   Man about the state of the stat
                              cboDeliveryMode.ListIndex = mudtClone.DeliveryMode
                              ' if key is not set, force "A"
                              If mudtClone.key = "" Then
                                     cboKey = "A"
                              Else
                                     cboKey = mudtClone.key
   ij
                              End If
304
                              If mudtClone.Nature = naPure Then
                                     optNature(0) = True
                                     optNature(1) = True
35
                              End If
                              sldTDEstimate = mudtClone.TDEstimate
                              chkRoute = mudtClone.IsRouted
                              chkCalcDifficulty = mudtClone.IsDifficultyCalculated
                              chkCalcDifficulty Click 'update screen accordingly
40
                              If mudtClone.IsDifficultyCalculated Then
                                     Select Case mudtFamily.Program
                                            Case prGRE
                                                  Set mudtGreDE = mudtClone.DiffEst
45
                                                  cboGREComp.ListIndex = mudtGreDE.Computation
```

	cboGRECog.ListIndex = mudtGreDE.Cognition cboGREConcept.ListIndex = mudtGreDE.Concept
	CreateDiffEst
	Case prGMAT
5	Set mudtGmatDE = mudtClone.DiffEst
	' nothing to load
	CreateDiffEst
	Case prSAT
1.0	' do nothing
10	End Select
	Else cboGREComp.ListIndex = 0
	cboGRECog.ListIndex = 0
	cboGREConcept.ListIndex = 0
15	End If
	mblnFormLoad = False
## 	End Sub
<u>4</u>] 2 0 1	D' (CI IOK CI'-LO
201	Private Sub cmdOK_Click()
իներ գիներ իներ գիրը կիրո Կերբ եւ են Կում եւ իր բումի	CreateProfile
47	
	Unload Me
2 5	End Sub
	Driveta Cub and Canaal Clieby
	Private Sub cmdCancel_Click()
	Unload Me
5 4	
	End Sub
30	Private Sub cboDomain Click()
	CreateProfile
	End Sub
	Drivinta Sub aba CDECoa Cliab
	Private Sub cboGRECog_Click()
	CreateProfile
35	End Sub

	Private Sub cboGREComp_Click()
	CreateProfile
	End Sub
5	Private Sub cboGREConcept_Click()
,	CreateProfile
	End Sub
	Private Sub cboKey_Click()
10	CreateProfile
	End Sub
	Private Sub optNature_Click(Index As Integer)
	CreateProfile
157	End Sub
1	Private Sub sldTDEstimate_Click()
	CreateProfile
2 0	End Sub
20 H.H.	Private Sub chkCalcDifficulty_Click()
24	fraPredDiff.Enabled = CBool(chkCalcDifficulty)
	fraGREDiff.Enabled = CBool(chkCalcDifficulty)
25	fraGMATDiff.Enabled = CBool(chkCalcDifficulty) lblGREComp.Enabled = CBool(chkCalcDifficulty)
	cboGREComp.Enabled = CBool(chkCalcDifficulty)
	lblGRECog.Enabled = CBool(chkCalcDifficulty)
	cboGRECog.Enabled = CBool(chkCalcDifficulty) lblConcept.Enabled = CBool(chkCalcDifficulty)
30	cboGREConcept.Enabled = CBool(chkCalcDifficulty)
	lblIRT.Enabled = CBool(chkCalcDifficulty)
	lbIIRTValue.Enabled = CBool(chkCalcDifficulty)
	lblPredDiff.Enabled = CBool(chkCalcDifficulty) lblPredFasy Enabled = CBool(chkCalcDifficulty)
	interedeasy enabled – C.BOOHCOKU AICLUHICHIIV)

	lblPredMed.Enabled = CBool(chkCalcDifficulty) lblPredDiff.Enabled = CBool(chkCalcDifficulty)
5	If chkCalcDifficulty Then CreateProfile End If
	End Sub
	Private Sub CreateProfile()
10	' don't do it if were still loading form If mblnFormLoad Then Exit Sub
15-4	mudtClone.Program = mudtFamily.Program mudtClone.Domain = cboDomain.ListIndex mudtClone.BatchID = txtBatchId mudtClone.DeliveryMode = cboDeliveryMode.ListIndex mudtClone.key = cboKey If optNature(0) = True Then
1\$.55 20.55	mudtClone.Nature = naPure
1 1	Else mudtClone.Nature = naReal
201	End If mudtClone.IsRouted = chkRoute
	mudtClone.TDEstimate = sldTDEstimate
14 4 Charles 25 Charles 20 Charle	mudtClone.IsDifficultyCalculated = chkCalcDifficulty
2 5	If chkCalcDifficulty Then
	CreateDiffEst End If
	End Sub
	Private Sub CreateDiffEst()
30	If mudtClone.IsDifficultyCalculated Then Set mudtDE = Nothing Select Case mudtFamily.Program Case prGRE
35	Set mudtGreDE = Nothing Set mudtGreDE = New GREDifficultyEstimate mudtGreDE.Domain = cboDomain.ListIndex mudtGreDE.Computation = cboGREComp.ListIndex

	mudtGreDE.Cognition = cboGRECog.ListIndex
	mudtGreDE.Concept = cboGREConcept.ListIndex
	mudtGreDE.key = cboKey
	If $optNature(0) = True Then$
5	mudtGreDE.Nature = naPure
	Else
	mudtGreDE.Nature = naReal
	End If
	mudtGreDE.ItemType = mudtFamily.ItemType
10	'attach this GRE DE to the clone
	mudtClone.DiffEst = mudtGreDE
	Set mudtDE = mudtGreDE
	SetPredDiffSlider
	Case prGMAT
15	Set mudtGmatDE = Nothing
	Set mudtGmatDE = New GMATDifficultyEstimate
	mudtGmatDE.Domain = cboDomain.ListIndex
	mudtGmatDE.key = cboKey
_ _ _ _ _ _ _ _ _ _ _	If optNature(0) = True Then
201	mudtGmatDE.Nature = naPure
201 U1 45 25	Else
UI **	mudtGmatDE.Nature = naReal
r ệ n Lin	End If
74. 74.	mudtGmatDE.ItemType = mudtFamily.ItemType mudtGmatDE.TDDiffEst = sldTDEstimate
25" 41"	'attach this GMAT DE to the clone
===	mudtClone.DiffEst = mudtGmatDE
	Set mudtDE = mudtGmatDE
#I	Set mudtDE = mudtGmatDE SetPredDiffSlider
10 m m m m m m m m m m m m m m m m m m m	Case prSAT
عَلِمُ ال	' do nothing
C)	End Select
L ead	Else ' opted not to calc difficulty
	mudtClone.DiffEst = Nothing
35	End If
	End Sub
	•
	Private Sub SetPredDiffSlider()
	,
	Dim dblIRT As Double
40	
	dblIRT = mudtDE.ComputeDifficulty
	lbIIRTValue = Format(dbIIRT, "0.#")

	Select Case mudthamily. Program
	Case prGRE
	If dblIRT < -1.001 Then
	sldDiffEstimate = 5
5	ElseIf dblIRT < -0.238 Then
	sldDiffEstimate = 4
	ElseIf dblIRT < 0.379 Then
	sldDiffEstimate = 3
	ElseIf dblIRT < 0.931 Then
10	sldDiffEstimate = 2
	Else
	sldDiffEstimate = 1
	End If
	Case prGMAT
15	If dblIRT < -0.919 Then
	sldDiffEstimate = 5
	ElseIf dblIRT < -0.093 Then
	sldDiffEstimate = 4
£1	ElseIf dblIRT < 0.565 Then
$2\overline{0}$	sldDiffEstimate = 3
7	ElseIf dblIRT < 1.197 Then
Ū1	sldDiffEstimate = 2
PA PAR	Else
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	sldDiffEstimate = 1
25 -	End If
W)	End Select
3 	
#"" to ""	End Sub
4.) F1	
ind Li	
F1	
="	

```
' frmDrag.frm
       VERSION 5.00
       Begin VB.Form frmDrag
                    = "Window drag'control"
         Caption
 5
         ClientHeight = 1005
         ClientLeft
                     = 60
         ClientTop
                     = 345
         ClientWidth = 3060
         LinkTopic
                     = "Form1"
         ScaleHeight
                      = 1005
10
                      = 3060
         ScaleWidth
         StartUpPosition = 2 'CenterScreen
         Begin VB.CommandButton Command2
                      = "Full Drag OFF"
          Caption
15
          Height
                     = 735
                    = 1560
          Left
          TabIndex
                       = 1
          Top
                     = 120
          Width
                     = 1215
         End
255
255
         Begin VB.CommandButton Command1
          Caption
                      = "Full Drag ON"
                     = 735
          Height
          Left
                    = 120
          TabIndex
                    = 120
          Top
 Width
                     = 1215
         End
       End
       Attribute VB Name = "frmDrag"
       Attribute VB GlobalNameSpace = False
       Attribute VB_Creatable = False
       Attribute VB PredeclaredId = True
       Attribute VB Exposed = False
35
       Option Explicit
       Private Declare Function SystemParametersInfo Lib "user32" _
         Alias "SystemParametersInfoA" (ByVal uAction As Long,
         ByVal uParam As Long, ByRef lpvParam As Any,
         ByVal fuWinIni As Long) As Long
40
       Private Const SPI_GETDRAGFULLWINDOWS = 38
       Private Const SPI SETDRAGFULLWINDOWS = 37
       Private Const SPIF_SENDWININICHANGE = 2
```

Dim result As Long 'Call API and check for successful call. If SystemParametersInfo(SPI GETDRAGFULLWINDOWS, 0&, result, 0&) \Leftrightarrow 0 Then 'Feature supported now check value of result. 5 If result = 0 Then IsFullWindowDragOn = False Else IsFullWindowDragOn = True 10 'Call failed, feature not supported. Else IsFullWindowDragOn = False End If 15 **End Function** Private Sub TurnOffFullWindowDrag() 20 Dim result As Long result = SystemParametersInfo(SPI_SETDRAGFULLWINDOWS, 0&, _ ByVal vbNullString, SPIF SENDWININICHANGE) End Sub Private Sub TurnOnFullWindowDrag() ļ., Dim result As Long result = SystemParametersInfo(SPI SETDRAGFULLWINDOWS, 1&, _ 25 ByVal vbNullString, SPIF SENDWININICHANGE) End Sub Private Sub Command1_Click() **TurnOnFullWindowDrag** 30 End Sub Private Sub Command2 Click()

Public Function IsFullWindowDragOn() As Boölean

End Sub

```
' frmIED.frm
       VERSION 5.00
       Begin VB.Form frmIED
                      = 1 'Fixed Single
         BorderStyle
 5
         Caption
                    = "TCA Installation"
         ClientHeight = 1185
                     = 45
         ClientLeft
         ClientTop
                     = 330
         ClientWidth = 2475
10
         LinkTopic
                     = "Form1"
                      = 0 'False
         MaxButton
                      = 0 'False
         MinButton
         ScaleHeight
                      = 1185
         ScaleWidth
                      = 2475
         StartUpPosition = 2 'CenterScreen
15
         Begin VB.CommandButton cmdOK
          Caption
                      = "OK"
          Height
                     = 375
          Left
                    = 600
          TabIndex
                       = 1
          Top
                     = 720
          Width
                      = 1215
         End
25<u>4</u>5
         Begin VB.Label Label1
          Caption
                      = "Setting IED files to read-only."
          Height
                     = 255
 Left
                    = 240
          TabIndex
                       = 0
                     = 240
          Top
          Width
                     = 2055
         End
       End
       Attribute VB Name = "frmIED"
       Attribute VB GlobalNameSpace = False
       Attribute VB Creatable = False
35
       Attribute VB PredeclaredId = True
       Attribute VB Exposed = False
       Option Explicit
       Private Sub cmdOK Click()
40
         Unload Me
```

End Sub

VBSCA -79-

Private Sub Form_Load()

Call Shell("attrib +r C:\tcs\working\dscbt.ied", vbHide)

Call Shell("attrib +r C:\tcs\working\qccbt.ied", vbHide)

Call Shell("attrib +r C:\tcs\working\qcppt.ied", vbHide)

Call Shell("attrib +r C:\tcs\working\ssmccbt.ied", vbHide)

Call Shell("attrib +r C:\tcs\working\ssmcppt.ied", vbHide)

```
' frmIndexedString.frm
       VERSION 5.00
      Object = "{6B7E6392-850A-101B-AFC0-4210102A8DA7}#1.3#0"; "COMCTL32.OCX"
      Begin VB.Form frmIndexedString
5
        BorderStyle
                   = 4 'Fixed ToolWindow
        ClientHeight = 2265
        ClientLeft
                   = 45
        ClientTop
                    = 285
        ClientWidth = 5835
        LinkTopic
                    = "Form1"
10
        LockControls = -1 'True
                     = 0 'False
        MaxButton
                    = 0 'False
        MinButton
                    = 2265
        ScaleHeight
15
        ScaleWidth
                     = 5835
        ShowInTaskbar = 0 'False
        StartUpPosition = 1 'CenterOwner
        Begin ComctlLib.ListView lvwIndexed
         Height
                    = 1815
         Left
                   = 120
         TabIndex
                     = 6
         Top
                    = 120
          Width
                    = 4215
          ExtentX
                    = 7435
25
          ExtentY
                     = 3201
          View
                    = 3
          Arrange
                     = 2
         LabelEdit
                     = 1
         MultiSelect = -1 'True
                      = -1 'True
         LabelWrap
         HideSelection = 0 'False
          Version
                     = 327682
         ForeColor
                     = -2147483640
         BackColor
                      = -2147483643
35
         BorderStyle = 1
          Appearance
                      = 1
         NumItems
         BeginProperty ColumnHeader(1) {0713E8C7-850A-101B-AFC0-4210102A8DA7}
           Key
40
           Object.Tag
                     = "Index"
           Text
                            = 529
           Object.Width
         EndProperty
         BeginProperty ColumnHeader(2) {0713E8C7-850A-101B-AFC0-4210102A8DA7}
```

```
SubItemIndex = 1
            Key
            Object.Tag
                      = "Value"
            Text
 5
            Object.Width
                             = 6174
          EndProperty
        End
        Begin VB.CommandButton cmdAdd
          Caption
                     = "Add"
10
          Height
                     = 255
          Left
                    = 120
                      = 5
          TabIndex
          ToolTipText = "Click here to add a value to the end of the list."
                     = 1900
          Top
15
          Width
                     = 975
        End
        Begin VB.CommandButton cmdInsert
          Caption
                      = "Insert"
          Height
                     = 255
          Left
                    = 1080
          TabIndex
                       = 4^{-1}
25.
          ToolTipText = "Click here to insert a value before the currently selected value."
                     = 1900
          Top
                     = 1095
          Width
        End
 41
        Begin VB.CommandButton cmdEdit
                     = "Edit"
          Caption
          Height
                     = 255
                    = 2160
          Left
          TabIndex
          ToolTipText = "Click here to edit the currently selected value."
          Top
                     = 1900
          Width
                     = 1095
        End
35
        Begin VB.CommandButton cmdRemove
                      = "Remove"
          Caption
          Height
                     = 255
          Left
                    = 3240
          TabIndex
          ToolTipText = "Click here to remove the selected value."
40
          Top
                     = 1900
          Width
                     = 1095
        End
        Begin VB.CommandButton cmdStrOK
```

= "OK"

Caption

45

```
= -1 'True
          Default
                      = 495
          Height
          Left
                     = 4440
                       = 0
          TabIndex
          ToolTipText = "Click here to save changes and return."
 5
          Top
                     = 120
          Width
                      = 1215
         End
         Begin VB.CommandButton cmdStrCancel
10
                      = "Cancel"
          Caption
                      = 495
          Height
          Left
                     = 4440
          TabIndex
                       = 1
          ToolTipText = "Click here to return without saving changes."
15
          Top
                     = 720
          Width
                      = 1215
         End
         Begin VB.Menu mnuIndexed
          Caption
                      = "Indexed"
                      = 0 'False
          Visible
          Begin VB.Menu mnuIndexedAdd
            Caption
                        = "Add"
 E. 4. C.
          End
          Begin VB.Menu mnuIndexedInsert
25
            Caption
                        = "Insert"
 41
          End
          Begin VB.Menu mnuIndexedEdit
            Caption
                        = "Edit"
          End
          Begin VB.Menu mnuIndexedRemove
            Caption
                        = "Remove"
          End
         End
       End
       Attribute VB Name = "frmIndexedString"
35
       Attribute VB GlobalNameSpace = False
       Attribute VB Creatable = False
       Attribute VB PredeclaredId = True
       Attribute VB Exposed = False
40
       Option Explicit
       Private mudtModel As Model
       Private mudtEF As EditFlags
       Private mstrVariableName As String
       Private mcolStrings As Collection
```

	Private mblnOK As Boolean
	Public Property Let Model(ByVal udtNewValue As Model)
	Set mudtModel = udtNewValue
	End Property
5	Public Property Let AddEditFlag(ByVal udtNewValue As EditFlags)
	mudtEF = udtNewValue
	End Property
	Public Property Let SubStringCollection(ByVal colNewValue As Collection)
10	Set mcolStrings = colNewValue
e a La	End Property
	Private Sub cmdAdd_Click()
₩1 ====================================	Call mnuIndexedAdd_Click
	End Sub
	Private Sub cmdEdit_Click()
20 -	Call mnuIndexedEdit_Click
2	End Sub
	Private Sub cmdInsert_Click()
	Call mnuIndexedInsert_Click
	End Sub
25	Private Sub cmdRemove_Click()
	Call mnuIndexedRemove_Click
	End Sub
	Private Sub Form Load()

```
Dim varS As Variant
          Dim lsiLI As ListItem
          Dim udtWAPI As New Win32API
 5
          'enable full row select
          Call udtWAPI.EnableListViewFullRowSelect(IvwIndexed)
          mblnOK = False
          frmIndexedString.Caption = "Editing substrings of string " & mstrVariableName
10
          If mudtEF = aeEdit Then
            With lvwIndexed
              For Each varS In mcolStrings
                Set lsiLI = .ListItems.Add
                UpdateListView
                lsiLI.SubItems(1) = varS
15
              Next varS
            End With
 ų,
          End If
 Ō١
 Uî
          'prevent changes if model is frozen
20:
          If mudtModel.IsFrozen Then
            cmdStrOK.Enabled = False
            cmdAdd.Enabled = False
            mnuIndexedAdd.Enabled = False
            cmdEdit.Caption = "Browse"
            mnuIndexedEdit.Caption = "Browse"
            cmdInsert.Enabled = False
            mnuIndexedInsert.Enabled = False
            cmdRemove.Enabled = False
30
            mnuIndexedRemove.Enabled = False
          End If
       End Sub
       Public Property Let VariableName(ByVal strNewValue As String)
          mstrVariableName = strNewValue
35
       End Property
       Public Property Get StringValue() As String
```

5	udtSS.Delimiter = Chr(STRING_DELIMITER) udtSS.StringCollection = mcolStrings StringValue = udtSS.StringValue
	End Property
	Public Property Get SubStringCollection() As Collection
10	Set SubStringCollection = mcolStrings
10	End Property
	Public Property Get OK() As Boolean
	OK = mblnOK
15	End Property
# 641 844 # 444 4344	Private Sub cmdStrOK_Click()
	Dim lsiItem As ListItem
15. If the content of the section of	Set mcolStrings = New Collection
26 and and are at a series	For Each lsiItem In lvwIndexed.ListItems Call mcolStrings.Add(lsiItem.SubItems(1)) Next lsiItem
	mblnOK = True
	Unload Me
25	End Sub
	Private Sub cmdStrCancel_Click()
	Unload Me
	End Sub
	Private Sub mnuIndexedAdd_Click()

With frmString

30

Dim udtSS As New SubString

```
' set the model
            .Model = mudtModel
            ' set the string
            .StringValue = ""
 5
            ' set var name
            .VariableName = mstrVariableName & "."
               & Trim(Str(IvwIndexed.ListItems.Count + 1))
            ' do it
            .Show vbModal
            If .OK = False Then Exit Sub
10
          End With
          Dim lsiNewItem As ListItem
          Set lsiNewItem = lvwIndexed.ListItems.Add
15
          UpdateListView
          lsiNewItem.SubItems(1) = frmString.StringValue
       End Sub
 C)
 IJ.
       Private Sub mnuIndexedEdit_Click()
 <u>f</u>î
 Ш
20:
          With frmString
            ' set the model
            .Model = mudtModel
            ' set the string
            .StringValue = lvwIndexed.SelectedItem.SubItems(1)
            ' set var name
            .VariableName = mstrVariableName & "."
               & Trim(Str(lvwIndexed.SelectedItem.Index))
            ' do it
            .Show vbModal
30
30
            If .OK = False Then Exit Sub
          End With
          lvwIndexed.SelectedItem.SubItems(1) = frmString.StringValue
       End Sub
       Private Sub mnuIndexedInsert Click()
35
          If IvwIndexed.SelectedItem Is Nothing Then Exit Sub
          With frmString
            ' set the Model
```

```
.Model = mudtModel
            ' set the string
            .StringValue = ""
            ' set var name
            .VariableName = mstrVariableName
 5
            ' do it
            .Show vbModal
            If .OK = False Then Exit Sub
          End With
10
          Dim lsiNewItem As ListItem
          Set lsiNewItem = lvwIndexed.ListItems.Add(lvwIndexed.SelectedItem.Index)
          UpdateListView
          lsiNewItem.SubItems(1) = frmString.StringValue
15
       End Sub
       Private Sub mnuIndexedRemove Click()
          If lvwIndexed.SelectedItem Is Nothing Then Exit Sub
 Ø
20
          Call lvwIndexed.ListItems.Remove(lvwIndexed.SelectedItem.Index)
          UpdateListView
       End Sub
       Private Sub UpdateListView()
          Dim intl As Integer
          For intI = 1 To lvwIndexed.ListItems.Count
            lvwIndexed.ListItems.Item(intI).Text = Str(intI)
          Next intI
```

```
' frmNew.frm
       VERSION 5.00
       Begin VB.Form frmNew
        BorderStyle
                     = 4 'Fixed ToolWindow
 5
        Caption
                    = "New family properties"
        ClientHeight = 1740
        ClientLeft
                    = 45
        ClientTop
                     = 285
        ClientWidth = 6240
10
        LinkTopic
                     = "Form1"
        LockControls = -1 'True
                     = 0 'False
        MaxButton
                     = 0 'False
        MinButton
        ScaleHeight
                     = 1740
15
        ScaleWidth
                     = 6240
        ShowInTaskbar = 0 'False
        StartUpPosition = 1 'CenterOwner
        Begin VB.CommandButton cmdCancel
          Cancel
                     = -1 'True
          Caption
                     = "Cancel"
2₫
          Height
                     = 495
          Left
                    = 4800
          TabIndex
                      = 9
          Top
                    = 720
25
          Width
                     = 1215
        End
        Begin VB.CommandButton cmdOK
                     = "OK"
          Caption
          Default
                     = -1 'True
          Height
                     = 495
          Left
                    = 4800
          TabIndex
                      = 8
          Top
                    = 120
          Width
                     = 1215
35
        End
        Begin VB.OptionButton optGeneric
          Caption
                     = "Non-generic"
          Height
                     = 195
          Index
                    = 1
40
                    = 3240
          Left
                      = 7
          TabIndex
          Top
                    = 1150
                     = 1455
          Width
```

End

```
Begin VB.OptionButton optGeneric
                      = "Generic"
          Caption
          Height
                     = 195
          Index
                     = 0
          Left
                    = 2280
 5
          TabIndex
                     = 6
                    = 1150
          Top
          Value
                     = -1 'True
          Width
                     = 975
10
         End
        Begin VB.ComboBox cboProximity
          Height
                     = 315
          ItemData
                      = "frmNew.frx":0000
          Left
                    = 2280
15
          List
                    = "frmNew.frx":000D
          Style
                    = 2 'Dropdown List
          TabIndex
                      = 4
          Top
                    = 360
          Width
                     = 1935
        End
        Begin VB.ComboBox cboItemType
 U1
          Height
                     = 315
 See Allen
                      = "frmNew.frx":0024
          ItemData
          Left
                    = 120
25
                    = "frmNew.frx":0031
          List
 ųj
          Style
                    = 2 'Dropdown List
          TabIndex
                      = 2
          Top
                    = 1080
          Width
                     = 1935
        End
        Begin VB.ComboBox cboProgram
          Height
                     = 315
          ItemData
                      = "frmNew.frx":0072
          Left
                    = 120
                    = "frmNew.frx":007F
35
          List
          Style
                    = 2 'Dropdown List
          TabIndex
                      = 0
          Top
                    = 360
          Width
                     = 1935
40
        End
        Begin VB.Label lbl
                     = "Variant proximity"
          Caption
          Height
                     = 255
          Left
                    = 2280
```

TabIndex

= 5

```
Top
                     = 120
          Width
                      = 1335
         End
         Begin VB.Label lblItemType
          Caption
                      = "Item type"
5
          Height
                      = 255
          Left
                     = 120
          TabIndex
                       = 3
          Top
                     = 840
10
          Width
                      = 1335
         End
         Begin VB.Label lblProgram
          Caption
                      = "Program"
          Height
                      = 255
15
          Left
                     = 120
          TabIndex
                       = 1
                     = 120
          Top
          Width
                      = 1335
         End
2Qj
       End
       Attribute VB Name = "frmNew"
25
       Attribute VB_GlobalNameSpace = False
       Attribute VB Creatable = False
       Attribute VB PredeclaredId = True
       Attribute VB Exposed = False
       Option Explicit
The state of
       Private mblnOK As Boolean
       Private mudtProgram As Program
j-1
       Private mudtItemType As ItemType
       Private mudtProximity As Proximity
3₫
       Private mblnGeneric As Boolean
       Private Sub Form_Load()
         mblnOK = False
35
         'init combo boxes
         cboProgram.ListIndex = 0
         cboItemType.ListIndex = 0
         cboProximity.ListIndex = 0
```

		Public Property Get OK() As Boolean
ı		OK = mblnOK
		End Property
	5	Public Property Get Program() As Program
ļ		Program = mudtProgram
		End Property
)		Public Property Get ItemType() As ItemType
	10	ItemType = mudtItemType
	10	End Property
)		Public Property Get Proximity() As Proximity
		Proximity = mudtProximity
		End Property
)	47 153 153 153 154 155 154 155 154 155 154 155 155 155	Public Property Get Generic() As Boolean
		Generic = mblnGeneric
)		End Property
	L	Private Sub cboProgram_Click()
	_	mudtProgram = cboProgram.ListIndex
)	. 20	End Sub
		Private Sub cboItemType_Click()
)		mudtItemType = cboItemType.ListIndex
		End Sub
		Private Sub cboProximity_Click()
)	25	mudtProximity = cboProximity.ListIndex

Private Sub optGeneric_Click(Index As Integer)

mblnGeneric = optGeneric(0)

End Sub

End Sub

5 Private Sub cmdOK_Click()

mblnOK = True

Unload Me

End Sub

10 Private Sub cmdCancel_Click()

Unload Me

```
' frmNewModel.frm
       VERSION 5.00
       Begin VB.Form frmNewFamily
        BorderStyle
                     = 4 'Fixed ToolWindow
                    = "New family"
 5
        Caption
        ClientHeight = 1350
        ClientLeft
                    = 45
        ClientTop
                     = 285
        ClientWidth = 4680
                     = "Form1"
10
        LinkTopic
        LockControls = -1 'True
                      = 0 'False
        MaxButton
                     = 0 'False
        MinButton
        ScaleHeight
                     = 1350
        ScaleWidth
15
                     = 4680
        ShowInTaskbar = 0 'False
        StartUpPosition = 1 'CenterOwner
        Begin VB.OptionButton optModelType
          Caption
                      = "Quantitative Comparision"
20
          Height
                     = 255
          Index
                     = 1
 The Res Am Car
                    = 480
          Left
                      = 4
          TabIndex
                    = 480
          Top
25
          Width
                     = 2535
        End
        Begin VB.OptionButton optModelType
                     = "Data Sufficiency"
          Caption
          Height
                     = 255
                     = 2
          Index
          Left
                    = 480
          TabIndex
                       = 3
          Top
                    = 720
          Width
                     = 2535
35
        End
        Begin VB.OptionButton optModelType
          Caption
                      = "Standard Multiple Choice"
          Height
                     = 255
          Index
                     = 0
                    = 480
40
          Left
          TabIndex
                      = 2
          Top
                    = 240
          Value
                     = -1 'True
                     = 2535
          Width
```

```
End
         Begin VB.CommandButton cmdCancel
                      = "Cancel"
          Caption
          Height
                      = 495
 5
          Left
                     = 3360
          TabIndex
          ToolTipText = "Click here to return without opening creating a new model."
                     = 720
          Top
          Width
                      = 1215
10
         End
         Begin VB.CommandButton cmdNewCreate
          Caption
                      = "Create"
                      = -1 'True
          Default
          Height
                      = 495
15
          Left
                     = 3360
          TabIndex
                       = 0
          ToolTipText = "Click here to create the new family."
          Top
                     = 120
          Width
                      = 1215
2Q)
         End
25 m
       End
       Attribute VB Name = "frmNewFamily"
       Attribute VB GlobalNameSpace = False
       Attribute VB Creatable = False
       Attribute VB PredeclaredId = True
       Attribute VB Exposed = False
       Option Explicit
 The first first
       Private mblnOK As Boolean
 j.
       'holds the item type
2
30]
       Private mudtItemType As ItemType
       Public Property Get OK() As Boolean
         OK = mblnOK
       End Property
35
       Public Property Get ItemType() As ItemType
         ItemType = mudtItemType
       End Property
```

Private Sub cmdNewCreate_Click()

mblnOK = True

5 Unload Me

End Sub

Private Sub cmdCancel_Click()

mblnOK = False

10 Unload Me

End Sub

Private Sub optModelType_Click(Index As Integer)

mudtItemType = Index

```
' frmProgram.frm
       VERSION 5.00
       Begin VB.Form frmProgram
                    = "Select the program"
         Caption
 5
         ClientHeight = 1350
         ClientLeft
                    = 60
         ClientTop
                    = 345
         ClientWidth = 3225
        LinkTopic
                     = "Form1"
10
         LockControls = -1 'True
         ScaleHeight = 1350
        ScaleWidth
                     = 3225
         StartUpPosition = 1 'CenterOwner
         Begin VB.OptionButton optProgram
                     = "SAT"
15
          Caption
          Height
                     = 195
          Index
                    = 2
          Left
                    = 240
          TabIndex
                      = 4
 Ú)
201
                    = 720
          Top
 Ľ١
          Width
                     = 1335
         End
         Begin VB.OptionButton optProgram
          Caption
                     = "GMAT"
25
          Height
                     = 195
          Index
                    = 1
          Left
                    = 240
                     = 3
          TabIndex
                    = 480
          Top
30
          Width
                     = 1335
         End
        Begin VB.OptionButton optProgram
          Caption
                     = "GRE"
                     = 195
          Height
                    = 0
35
          Index
          Left
                    = 240
          TabIndex
                      = 2
                    = 240
          Top
          Value
                    = -1 'True
40
          Width
                     = 1335
         End
        Begin VB.CommandButton cmdCancel
                     = "Cancel"
          Caption
          Height
                     = 495
```

```
Left
                      = 1920
           TabIndex
                        = 1
           ToolTipText = "Click here to return."
                      = 720
 5
           Width
                       = 1215
         End
         Begin VB.CommandButton cmdOK
                       = "OK"
           Caption
                       = 495
           Height
                      = 1920
10
           Left
           TabIndex
                        = 0
           ToolTipText
                         = "Click here to save the currently selected program and return."
           Top
           Width
                       = 1215
15
         End
       End
       Attribute VB Name = "frmProgram"
       Attribute VB GlobalNameSpace = False
       Attribute VB Creatable = False
       Attribute VB PredeclaredId = True
       Attribute VB Exposed = False
 Man allen Han allen Care Carls
       Option Explicit
       Private mblnOK As Boolean
       Private mudtProgram As Program
       Public Property Get OK() As Boolean
          OK = mblnOK
       End Property
       Public Property Get Program() As Program
30
          Program = mudtProgram
       End Property
       Private Sub cmdOK Click()
          mblnOK = True
35
```

Unload Me

```
End Sub

Private Sub cmdCancel_Click()

mblnOK = False

Unload Me

End Sub

Private Sub optProgram_Click(Index As Integer)

mudtProgram = Index
```

```
' frmProgress.frm
       VERSION 5.00
       Object = "{6B7E6392-850A-101B-AFC0-4210102A8DA7}#1.2#0"; "COMCTL32.OCX"
       Begin VB.Form frmProgress
 5
         BorderStyle
                     = 1 'Fixed Single
         ClientHeight = 1110
        ClientLeft
                    = 15
         ClientTop
                     = 15
         ClientWidth = 4500
10
         ClipControls = 0 'False
         ControlBox
                      = 0 'False
         LinkTopic
                     = "Form1"
         LockControls = -1 'True
        MaxButton
                     = 0 'False
                      = 0 'False
15
         MinButton
         ScaleHeight
                     = 1110
         ScaleWidth
                      = 4500
         StartUpPosition = 2 'CenterScreen
 Begin ComctlLib.ProgressBar prbProgressBar
 T,
20
          Height
                     = 255
 IJ.
          Left
                   = 240
          TabIndex
                      = 0
                    = 600
          Top
          Width
                     = 3975
25
          ExtentX
                      = 7011
          _ExtentY
                      = 450
Version
                      = 327682
 T. T.
          Appearance
                       = 1
                     = 500
          Max
3<u>0</u> j
         End
         Begin VB.Label lblProgress
                       = 2 'Center
          Alignment
          BeginProperty Font
           Name
                       = "MS Sans Serif"
35
            Size
                      = 8.25
                       = 0
            Charset
                       = 700
            Weight
           Underline
                        = 0 'False
                     = 0 'False
           Italic
40
            Strikethrough = 0 'False
          EndProperty
          Height
                     = 255
          Left
                    = 240
          TabIndex
                     = 1
```

Top = 240
Width = 3855
End
End

5 Attribute VB_Name = "frmProgress"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False

Option Explicit

```
' frmProlog.frm
       VERSION 5.00
       Begin VB.Form frmProlog
                      = 5 'Sizable ToolWindow
         BorderStyle
 5
         ClientHeight = 900
         ClientLeft
                     = 2775
         ClientTop
                      = 3720
                      = 4440
         ClientWidth
         LinkTopic
                      = "Form1"
10
         LockControls = -1 'True
         MaxButton
                    = 0 'False
                      = 0 'False
         MinButton
         ScaleHeight
                      = 900
                      = 4440
         ScaleWidth
15
         ShowInTaskbar = 0 'False
         StartUpPosition = 2 'CenterScreen
         Begin VB.CommandButton cmdAbort
           Caption
                      = "Abort"
          Default
                      = -1 'True
 ű
201
                      = 495
          Height
 U
          Left
                     = 3120
 W. Jan
           TabIndex
                       = 0
                     = 120
           Top
           Width
                      = 1215
25<sup>-1</sup>
         End
         Begin VB.Label lblProlog
          Height
                      = 495
          Left
                     = 120
           TabIndex
                       = 1
                     = 120
           Top
           Width
                      = 2655
         End
       End
       Attribute VB Name = "frmProlog"
       Attribute VB GlobalNameSpace = False
35
       Attribute VB Creatable = False
       Attribute VB PredeclaredId = True
       Attribute VB Exposed = False
       Option Explicit
       Private mblnAbort As Boolean
40
```

Public Property Get Abort() As Boolean

Public Sub Kill()

5 Unload Me

End Sub

Private Sub Form_Load()

mblnAbort = False

10 End Sub

Private Sub cmdAbort_Click()

mblnAbort = True Unload Me

End Sub

```
' frmSplash.frm
       VERSION 5.00
       Begin VB.Form frmSplash
         BorderStyle
                      = 3 'Fixed Dialog
 5
         ClientHeight = 4245
         ClientLeft
                     = 255
         ClientTop
                     = 1410
         ClientWidth
                     = 7380
         ClipControls = 0 'False
10
         ControlBox
                      = 0 'False
                   = "frmSplash.frx":0000
         Icon
         KeyPreview
                      = -1 'True
         LinkTopic
                     = "Form2"
         LockControls = -1 'True
15
         MaxButton
                      = 0 'False
                      = 0 'False
         MinButton
         ScaleHeight
                      = 4245
                      = 7380
         ScaleWidth
         ShowInTaskbar = 0 'False
 4)
         StartUpPosition = 2 'CenterScreen
20
25
         Begin VB.Frame fraSplash
                      = 4050
          Height
          Left
                    = 120
          TabIndex
                       = 0
          Top
                     = 60
          Width
                     = 7080
 m. m.
          Begin VB.Image imgLogo
            BorderStyle = 1 'Fixed Single
            Height
                       = 780
3⊕
            Left
                      = 600
            Picture
                       = "frmSplash.frx":000C
 Top
                      = 720
            Width
                       = 1275
          End
35
          Begin VB.Label lblCopyright
                        = "Copyright 1999"
            Caption
            BeginProperty Font
             Name
                         = "Arial"
              Size
                        = 8.25
40
              Charset
                         = 0
                         = 400
              Weight
                          = 0 'False
              Underline
                       = 0 'False
              Italic
              Strikethrough = 0 'False
```

```
EndProperty
            Height
                       = 255
            Left
                      = 4560
            TabIndex
                         = 3
 5
            Top
                       = 3480
            Width
                       = 2415
          End
          Begin VB.Label lblCompany
            Caption
                        = "Educational Testing Service"
            BeginProperty Font
10
              Name
                          = "Arial"
              Size
                        = 8.25
              Charset
                         = 0
              Weight
                          = 400
15
              Underline
                          = 0 'False
                       = 0 'False
              Italic
              Strikethrough = 0 'False
            EndProperty
            Height
                       = 255
201
                      = 4560
            Left
            TabIndex
                         = 2
 Ø١
 F. S. C.
            Top
                       = 3720
            Width
                       = 2415
          End
25
          Begin VB.Label lblWarning
 47
            Caption
                        = "Proprietary and Confidential"
            BeginProperty Font
3Q1
              Name
                         = "Arial"
                        = 9.75
              Size
                         = 0
              Charset
              Weight
                          = 700
              Underline
                          = 0 'False
                       = 0 'False
              Italic
              Strikethrough = 0 'False
35
            EndProperty
            Height
                       = 315
                      = 240
            Left
            TabIndex
                         = 1
                      = 3600
            Top
40
            Width
                       = 2775
          End
          Begin VB.Label lblVersion
                         = 1 'Right Justify
            Alignment
                        = -1 'True
            AutoSize
```

= "Version 1.25"

45

Caption

```
BeginProperty Font
                         = "Arial"
              Name
              Size
                        = 12
                         = 0
              Charset
                         = 700
 5
              Weight
              Underline
                          = 0 'False
              Italic
                       = 0 'False
              Strikethrough = 0 'False
            EndProperty
10
            Height
                       = 285
            Left
                      = 5265
                        = 4
            TabIndex
                      = 2880
            Top
            Width
                       = 1410
15
          End
          Begin VB.Label lblProductName
            AutoSize
                        = -1 'True
            Caption
                        = "Assistant"
            BeginProperty Font
                         = "Arial"
              Name
              Size
                        = 48
 Man Han Maria
              Charset
                         = 0
                         = 700
              Weight
              Underline
                          = 0 'False
25
              Italic
                       = 0 'False
 1
              Strikethrough = 0 'False
            EndProperty
            Height
                       = 1125
            Left
                      = 1440
            TabIndex
                        = 6
                       = 1560
            Top
            Width
                       = 4320
          Begin VB.Label lblCompanyProduct
                        = -1 'True
35
            AutoSize
                        = "Test Creation"
            Caption
            BeginProperty Font
                         = "Arial"
              Name
              Size
                        = 1.8
40
                         = 0
              Charset
                         = 700
              Weight
                          = 0 'False
              Underline
              Italic
                       = 0 'False
              Strikethrough = 0 'False
```

EndProperty

```
Height
                                 = 435
                 Left
                               = 2400
                 TabIndex
                                   = 5
                                = 1080
                 Top
 5
                 Width
                                 = 2400
              End
            End
          End
          Attribute VB_Name = "frmSplash"
Attribute VB_GlobalNameSpace = False
10
          Attribute VB_Creatable = False
          Attribute VB_PredeclaredId = True
          Attribute VB_Exposed = False
          Option Explicit
15
          Public Sub UnloadMe()
             Unload Me
 The show the sheet than the little little in the stand
          End Sub
 Hall Hall Har Hall Har Hall Hall
```

```
'SetPrecision.frm
       VERSION 5.00
       Begin VB.Form frmSetPrecision
         BorderStyle = 4 'Fixed ToolWindow
 5
         Caption
                  = "Set Precision"
         ClientHeight = 1965
         ClientLeft
                    = 45
         ClientTop
                     = 285
         ClientWidth = 3540
10
         LinkTopic
                     = "Form1"
                      = 0 'False
         MaxButton
                      = 0 'False
         MinButton
         ScaleHeight
                      = 1965
         ScaleWidth
                      = 3540
15
         ShowInTaskbar = 0 'False
         StartUpPosition = 2 'CenterScreen
         Begin VB.CommandButton cmdSetPrecisionDefault
                      = "Default"
          Caption
          Height
                      = 495
          Left
                    = 2160
25
          TabIndex
                       = 3
          ToolTipText = "Click here to return to the default value for precision."
          Top
                     = 1320
          Width
                      = 1215
         End
        Begin VB.CommandButton cmdSetPrecisionOK
 The state of the
          Caption
                      = "OK"
                      = -1 'True
          Default
          Height
                     = 495
          Left
                    = 2160
          TabIndex
          ToolTipText = "Click here to save the displayed value."
          Top
                     = 120
          Width
                     = 1215
35
         End
         Begin VB.CommandButton cmdSetPrecisionCancel
          Caption
                      = "Cancel"
          Height
                      = 495
          Left
                    = 2160
40
          TabIndex
          ToolTipText
                        = "Click here to return without saving any changes to precision."
                     = 720
          Top
          Width
                     = 1215
```

```
Begin VB.TextBox txtPrecision
           Height
                       = 315
           Left
                      = 120
           TabIndex
                        = 0
                      = ".1"
 5
           Text
           Top
                      = 120
           Width
                       = 1815
         End
       End
       Attribute VB Name = "frmSetPrecision"
10
       Attribute VB GlobalNameSpace = False
       Attribute VB Creatable = False
        Attribute VB PredeclaredId = True
       Attribute VB Exposed = False
       Option Explicit
15
       Private Sub cmdSetPrecisionCancel_Click()
          Unload Me
 ű
       End Sub
 <u>a</u>i
 IJſ
20
       Private Sub cmdSetPrecisionDefault Click()
 41
 txtPrecision = ".001"
       End Sub
       Private Sub cmdSetPrecisionOK Click()
          frmTCA.Precision = txtPrecision
          Unload Me
       End Sub
       Private Sub Form Load()
30
          txtPrecision = frmTCA.Precision
       End Sub
       Private Sub txtPrecision GotFocus()
          ' Automatically select all text when TextBox gets focus
          Call txtSelectAll(txtPrecision)
35
```

VBSCA -110-

```
'String.frm
       VERSION 5.00
       Begin VB.Form frmString
                     = 4 'Fixed ToolWindow
         BorderStyle
 5
         ClientHeight = 2265
         ClientLeft
                     = 45
         ClientTop
                     = 285
        ClientWidth = 5835
                     = "Form1"
         LinkTopic
10
        LockControls = -1 'True
        MaxButton
                    = 0 'False
                      = 0 'False
         MinButton
         ScaleHeight
                     = 2265
         ScaleWidth
                      = 5835
         ShowInTaskbar = 0 'False
15
         StartUpPosition = 1 'CenterOwner
         Begin VB.CommandButton cmdStrOK
          Caption
                      = "OK"
                     = -1 'True
          Default
                     = 495
201
          Height
As West allen a then
          Left
                    = 4440
          TabIndex
                       = 1
          ToolTipText
                        = "Click here to save changes and return."
                     = 120
          Top
25
          Width
                     = 1215
         End
Begin VB.CommandButton cmdStrCancel
                      = "Cancel"
          Caption
          Height
                     = 495
3₽ĵ
          Left
                    = 4440
          TabIndex
                       = 2
          ToolTipText = "Click here to return without saving changes."
          Top
                     = 720
          Width
                     = 1215
35
         End
         Begin VB.TextBox txtString
          Height
                     = 315
                    = 240
          Left
          TabIndex
                       = 0
40
          Top
                     = 480
          Width
                     = 3975
         End
       End
       Attribute VB Name = "frmString"
```

5	Attribute VB_GlobalNameSpace = False Attribute VB_Creatable = False Attribute VB_PredeclaredId = True Attribute VB_Exposed = False Option Explicit
	Private mudtModel As Model Private mstrVariableName As String Private mstrStringValue As String Private mblnOK As Boolean
10	Public Property Let Model(ByVal udtNewValue As Model)
	Set mudtModel = udtNewValue
	End Property
	Public Property Let VariableName(ByVal strNewValue As String)
157	mstrVariableName = strNewValue
	End Property
	Public Property Let StringValue(ByVal strNewValue As String)
7. 1.]	mstrStringValue = strNewValue
	End Property
	Public Property Get StringValue() As String
	StringValue = mstrStringValue
25	End Property
	Public Property Get OK() As Boolean
	OK = mblnOK
	End Property
30	Private Sub Form_Load()

mblnOK = False

```
frmString.Caption = "Editing string " & mstrVariableName
          txtString = mstrStringValue
 5
          If mudtModel.IsFrozen Then
            cmdStrOK.Enabled = False
          End If
       End Sub
10
       Private Sub cmdStrOK_Click()
          mblnOK = True
          StringValue = txtString
          Unload Me
15
       End Sub
       Private Sub cmdStrCancel_Click()
Unload Me
       End Sub
       Private Sub txtString_GotFocus()
          ' Automatically select all text when TextBox gets focus
          Call txtSelectAll(txtString)
```

End Sub

```
'TCA.FRM
       VERSION 5.00
       Object = "{6B7E6392-850A-101B-AFC0-4210102A8DA7}#1.3#0"; "COMCTL32.OCX"
       Object = "{BDC217C8-ED16-11CD-956C-0000C04E4C0A}#1.1#0"; "TABCTL32.OCX"
       Object = "{F9043C88-F6F2-101A-A3C9-08002B2F49FB}#1.2#0"; "COMDLG32.OCX"
 5
       Begin VB.Form frmTCA
                   = "ETS Test Creation Assistant"
        Caption
        ClientHeight = 8310
        ClientLeft
                   = 165
10
        ClientTop
                    = 735
        ClientWidth = 11400
                    = "Form1"
        LinkTopic
        LockControls = -1 'True
        ScaleHeight = 8310
15
                     = 11400
        ScaleWidth
        StartUpPosition = 3 'Windows Default
        Begin VB.Frame frmDummy
                     = "Common dialog anchor"
          Caption
          Height
                    = 855
          Left
                   = 2640
          TabIndex
                     = 3
 = 2280
          Top
          Visible
                    = 0 'False
          Width
                    = 2055
25
          Begin MSComDlg.CommonDialog cdlCD
           Left
                     = 120
           Top
                     = 240
 11,
           _ExtentX
                       = 847
           ExtentY
                       = 847
           Version
                       = 393216
          End
        End
        Begin VB.Frame fraWord
          Height
                    = 8535
          Left
35
                   = 120
                     = 1
          TabIndex
                    = 0
          Top
                    = 6255
          Width
        End
40
        Begin TabDlg.SSTab sstMainTab
          Height
                    = 8535
          Left
                   = 6480
          TabIndex
                      = 0
          Top
                    = 0
```

```
= 5655
          Width
                       = 9975
          ExtentX
           ExtentY
                       = 15055
                       = 393216
           Version
5
                       = 520
          TabHeight
          BeginProperty Font {0BE35203-8F91-11CE-9DE3-00AA004BB851}
                        = "MS Sans Serif"
            Name
                      = 8.25
            Size
                       = 0
            Charset
10
            Weight
                        = 400
                        = 0 'False
            Underline
                      = 0 'False
            Italic
            Strikethrough = 0 'False
          EndProperty
          TabCaption(0) = "Family Overview"
15
          TabPicture(0) = "TCA.frx":0000
          Tab(0).ControlEnabled= -1 'True
          Tab(0).Control(0)= "lblFamily"
          Tab(0).Control(0).Enabled= 0 'False
          Tab(0).Control(1) = "imlI"
          Tab(0).Control(1).Enabled= 0 'False
          Tab(0).Control(2)= "lblDummy"
 Tab(0).Control(2).Enabled= 0 'False
          Tab(0).Control(3)= "lblAccepted"
25
          Tab(0).Control(3).Enabled= 0 'False
 Ą,
          Tab(0).Control(4)= "lstAccepted"
          Tab(0).Control(4).Enabled= 0 'False
          Tab(0).Control(5)= "txtVariablize"
          Tab(0).Control(5).Enabled= 0 'False
          Tab(0).Control(6)= "treModels"
          Tab(0).Control(6).Enabled= 0 'False
          Tab(0).Control(7)= "cmdSetAttributes"
          Tab(0).Control(7).Enabled= 0 'False
          Tab(0).Control(8) = "lstDummy"
          Tab(0).Control(8).Enabled= 0 'False
35
          Tab(0).Control(9)= "cmdDone"
          Tab(0).Control(9).Enabled= 0 'False
          Tab(0).Control(10)= "cmdPrintBatch"
          Tab(0).Control(10).Enabled= 0 'False
          Tab(0).Control(11)= "cmdTreeExtend"
40
          Tab(0).Control(11).Enabled= 0 'False
          Tab(0).Control(12)= "cmdTreeRemove"
          Tab(0).Control(12).Enabled= 0 'False
          Tab(0).Control(13)= "cmdAcceptedPaste"
```

Tab(0).Control(13).Enabled= 0 'False

```
Tab(0).Control(14)= "cmdAcceptedCopy"
           Tab(0).Control(14).Enabled= 0 'False
           Tab(0).Control(15)= "cmdAcceptedEdit"
           Tab(0).Control(15).Enabled= 0 'False
 5
           Tab(0).ControlCount= 16
           TabCaption(1) = "Model Workshop"
           TabPicture(1) = "TCA.frx":001C
           Tab(1).ControlEnabled= 0 'False
           Tab(1).Control(0)= "lblVariables"
           Tab(1).Control(1)= "lblCloningConstraints"
10
           Tab(1).Control(2)= "lblDistractor"
           Tab(1).Control(3)= "cmdExportConstraints"
           Tab(1).Control(4)= "cmdImportConstraints"
           Tab(1).Control(5) = "cmdSaveModel"
           Tab(1).Control(6)= "cmdTestAll"
15
           Tab(1).Control(7)= "lstConstraints(1)"
           Tab(1).Control(8)= "cmdVariableAdd"
           Tab(1).Control(9)= "cmdVariableEdit"
           Tab(1).Control(10)= "cmdVariableRemove"
           Tab(1).Control(11)= "cmdVariableTest"
           Tab(1).Control(12)= "cmdConstraintAdd(0)"
 IJ
           Tab(1).Control(13)= "cmdConstraintEdit(0)"
           Tab(1).Control(14)= "cmdConstraintRemove(0)"
           Tab(1).Control(15)= "cmdConstraintTest(0)"
25
           Tab(1).Control(16)= "cmdConstraintAdd(1)"
 ij
           Tab(1).Control(17)= "cmdConstraintEdit(1)"
           Tab(1).Control(18)= "cmdConstraintRemove(1)"
           Tab(1).Control(19) = "cmdConstraintTest(1)"
           Tab(1).Control(20)= "cmdPrintConstraints"
           Tab(1).Control(21)= "lstConstraints(0)"
           Tab(1).Control(22)= "lstVariables"
           Tab(1).Control(23)= "cmdComments"
           Tab(1).ControlCount= 24
           TabCaption(2) = "Generate Variants"
35
           TabPicture(2) = "TCA.frx":0038
           Tab(2).ControlEnabled= 0 'False
           Tab(2).Control(0)= "cmdDispMakeModel"
           Tab(2).Control(1)= "cmdDispDiscard"
           Tab(2).Control(2)= "cmdDispDefer"
           Tab(2).Control(3)= "cmdDispAccept"
40
           Tab(2).Control(4)= "sldDifference"
           Tab(2).Control(5)= "lstDisposition"
           Tab(2).Control(6)= "cmdPrintVariants"
           Tab(2).Control(7)= "cmdDisplayModel"
           Tab(2).Control(8)= "txtNum2Generate"
45
```

```
Tab(2).Control(9)= "cmdGenerate"
           Tab(2).Control(10) = "lblDiff"
           Tab(2).Control(11)= "Label1"
           Tab(2).Control(12)= "lblMed"
 5
           Tab(2).Control(13)= "lblLow"
           Tab(2).Control(14)= "lblVariants"
           Tab(2).Control(15)= "LblNumVariants"
           Tab(2).ControlCount= 16
           Begin VB.CommandButton cmdComments
10
            Caption '
                        = "Comments"
            Height
                        = 495
            Left
                       = -70680
                         = 58
            TabIndex
            ToolTipText = "Click here to print all variables and constraints."
15
            Top
                       = 3720
            Width
                        = 1215
           End
           Begin VB.ListBox lstVariables
            DragIcon
                         = "TCA.frx":0054
            Height
                        = 1635
                         = "TCA.frx":035E
            ItemData
            Left
                       = -74760
                      = "TCA.frx":0360
            List
            Style
                       = 1 'Checkbox
            TabIndex
                         = 57
            ToolTipText
                          = "Left button click to select a constraint. Then right button click for
       constraint options."
            Top
                       = 720
            Width
                        = 3855
           End
          Begin VB.ListBox lstConstraints
            DragIcon
                         = "TCA.frx":0362
            Height
                        = 1635
            Index
                        = 0
                         = "TCA.frx":066C
35
            ItemData
            Left
                       = -74760
            List
                      = "TCA.frx":066E
            Style
                       = 1 'Checkbox
            TabIndex
40
            ToolTipText
                          = "Left button click to select a constraint. Then right button click for
       constraint options."
            Top
                       = 3120
            Width
                        = 3855
          End
```

Begin VB.CommandButton cmdAcceptedEdit

```
= "Edit Profile"
            Caption
            Height
                        = 255
            Left
                       = 240
                         = 54
            TabIndex
            ToolTipText = "Click here to edit the profile of the selected variant."
 5
            Top
                       = 7300
                        = 1335
            Width
           End
           Begin VB.CommandButton cmdAcceptedCopy
                        = "Copy Profile"
10
            Caption
            Height
                        = 255
            Left
                       = 1560
                         = 53
            TabIndex
            ToolTipText = "Click here to copy the profile of the selected variant."
15
                       = 7300
            Top
            Width
                        = 1335
          End
          Begin VB.CommandButton cmdAcceptedPaste
                        = "Paste Profile"
            Caption
            Height
                        = 255
            Left
                       = 2880
 Mary Steer Com
            TabIndex
                         = 52
            ToolTipText = "Click here to paste a profile onto the currently selected variants."
            Top
                       = 7300
25
                        = 1215
            Width
 Ü
          End
          Begin VB.CommandButton cmdPrintConstraints
            Caption
                        = "Print Constraints"
            Height
                        = 495
            Left
                       = -70680
            TabIndex
                         = 51
            ToolTipText = "Click here to print all variables and constraints."
            Top
                       = 3120
            Width
                        = 1215
35
          End
          Begin VB.CommandButton cmdDispMakeModel
                       = "Create Mdl."
            Caption
            Height
                        = 255
                       = -71880
            Left
40
            TabIndex
                         = 50
                          = "Click here to create new children of the active model using the
            ToolTipText
       currently selected variants."
                       = 6120
            Top
                        = 975
            Width
```

```
Begin VB.CommandButton cmdDispDiscard
            Caption
                        = "Discard"
            Height
                       = 255
                      = -72840
            Left
 5
                         = 49
            TabIndex
            ToolTipText = "Click here to discard the currently selected variants."
                       = 6120
            Top
            Width
                       = 975
10
          Begin VB.CommandButton cmdDispDefer
                        = "Defer"
            Caption
            Height
                       = 255
            Left
                      = -73800
            TabIndex
                        = 48
15
            ToolTipText = "Click here to defer the currently selected variants."
                       = 6120
            Top
                       = 975
            Width
          End
          Begin VB.CommandButton cmdDispAccept
            Caption
                        = "Accept"
            Height
                       = 255
 Ú1
            Left
                      = -74760
 Mr. Hay
            TabIndex
                        = 47
            ToolTipText = "Click here to accept the currently selected variants."
25
            Top
                      = 6120
 ij,
            Width
                       = 975
          End
          Begin VB.CommandButton cmdTreeRemove
                       = "Remove"
            Caption
            Height
                       = 255
            Left
                      = 2160
                       = 46
            TabIndex
            ToolTipText = "Click here to remove a model."
            Top
                      = 3720
                       = 1935
35
            Width
          End
          Begin VB.CommandButton cmdTreeExtend
            Caption
                       = "Extend"
            Height
                       = 255
40
            Left
                      = 240
            TabIndex
            ToolTipText = "Click here to create a new child of the selected model."
                      = 3720
            Top
            Width
                       = 1935
```

```
Begin VB.CommandButton cmdConstraintTest
                        = "Test"
            Caption
            Height
                       = .255
            Index
                       = 1
            Left
 5
                      = -71880
                         = 44
            TabIndex
                          = "Click here to test all enabled variables and distractor constraints."
            ToolTipText
                       = 7200
            Top
            Width
                       = 975
10
           End
          Begin VB.CommandButton cmdConstraintRemove
                        = "Remove"
            Caption
            Height
                       = 255
            Index
                       = 1
                      = -72840
15
            Left
            TabIndex
                         = 43
            ToolTipText = "Click here to remove a distractor constraint."
            Top
                       = 7200
                       = 975
            Width
20
          End
          Begin VB.CommandButton cmdConstraintEdit
 <u>O</u>1
 U
            Caption
                        = "Edit"
 Here allow
            Height
                       = 255
            Index
                       = 1
                      = -73800
            Left
 41
            TabIndex
                         = 42
            ToolTipText = "Click here to edit the currently selected distractor constraint."
            Top
                       = 7200
            Width
                       = 975
          End
          Begin VB.CommandButton cmdConstraintAdd
            Caption
                        = "Add"
            Height
                       = 255
            Index
                       = 1
35
            Left
                      = -74760
            TabIndex
                         = 41
            ToolTipText = "Click here to add a distractor constraint."
            Top
                       = 7200
            Width
                       = 975
40
          End
          Begin VB.CommandButton cmdConstraintTest
                        = "Test"
            Caption
            Height
                       = 255
            Index
                       = 0
45
            Left
                      = -71880
```

```
TabIndex
            ToolTipText = "Click here to test all enabled variables and variation constraints."
            Top
                      = 4800
                       = 975
            Width
 5
          End
          Begin VB.CommandButton cmdConstraintRemove
                       = "Remove"
                       = 255
            Height
            Index
                       = 0
                      = -72840
10
            Left
            TabIndex = 39
                         = "Click here to remove the currently selected variation constraint."
            ToolTipText
            Top
                      = 4800
            Width
                       = 975
15
          End
          Begin VB.CommandButton cmdConstraintEdit
                       = "Edit"
            Caption
            Height
                       = 255
            Index
                       = 0
            Left
                      = -73800
                        = 38
            TabIndex
            ToolTipText = "Click here to edit the currently selected variation constraint."
 = 4800
            Top.
            Width
                       = 975
25
          End
          Begin VB.CommandButton cmdConstraintAdd
                       = "Add"
            Caption
            Height
                       = 255
            Index
                       = 0
                      = -74760
            Left
                        = 37
            TabIndex
                         = "Click here to add a variation constraint."
            ToolTipText
            Top
                      = 4800
            Width
                       = 975
35
          End
          Begin VB.CommandButton cmdVariableTest
                       = "Test"
            Caption
            Height
                       = 255
                      = -71880
            Left
40
                        = 36
            TabIndex
            ToolTipText
                         = "Click here to test all enabled variables."
                      = 2400
            Top
                       = 975
            Width
          End
```

VBSCA -121-

Begin VB.CommandButton cmdVariableRemove

```
Caption
                        = "Remove"
                       = 255
            Height
                      = -72840
            Left
            TabIndex
                         = 35
                          = "Click here to remove the currently selected variable."
 5
            ToolTipText
            Top
                      = 2400
                       = 975
            Width
          End
          Begin VB.CommandButton cmdVariableEdit
            Caption
                        = "Edit"
10
            Height
                       = 255
            Left
                      = -73800
                         = 34
            TabIndex
            ToolTipText = "Click here to edit the currently selected variable."
15
            Top
                      = 2400
            Width
                       = 975
          End
          Begin VB.CommandButton cmdVariableAdd
            Caption
                        = "Add"
            Height
                       = 255
            Left
                      = -74760
 Mr. Bu Ca
            TabIndex
                         = 33
            ToolTipText = "Click here to add a variable."
            Top
                      = 2400
25
            Width
                       = 975
 ű,
          End
          Begin VB.CommandButton cmdPrintBatch
            Caption
                        = "Print All"
            Height
                       = 495
                      = 4320
            Left
                         = 31
            TabIndex
                          = "Click here to print all variants."
            ToolTipText
            Top
                      = 4200
                       = 1215
            Width
35
          End
          Begin VB.CommandButton cmdDone
            Caption
                        = "Done"
            Height
                       = 495
            Left
                      = 4320
40
                        = 29
            TabIndex
            ToolTipText
                          = "Click here when you are done with this family and are ready to send it
       back to TCS."
            Top
                      = 1320
            Width
                       = 1215
```

```
Begin ComctlLib.Slider sldDifference
            Height
                        = 255
            Left
                      = -73440
                         = 24
            TabIndex
                          = "Select the degree of randomization desired."
 5
            ToolTipText
            Top
                       = 1140
            Width
                        = 1935
            ExtentX
                         = 3413
            ExtentY
                         = 450
10
             Version
                        = 327682
                       = 2
            Max
            SelStart
                       = 2
            Value
                       = 2
          End
15
          Begin VB.ListBox lstDisposition
                       = 3570
            Height
                         = "TCA.frx":0670
            ItemData
            Left
                      = -74760
                      = "TCA.frx":0672
            List
            MultiSelect = 2 'Extended
                         = 21.
            TabIndex
 Mr. Jen C...
            ToolTipText
                          = "Left button click to select a variant. Then right button click for variant
       options."
            Top
                       = 2520
25
            Width
                       = 3855
          End
          Begin VB.CommandButton cmdPrintVariants
            Caption
                        = "Print All"
                       = 495
            Height
            Left
                      = -70680
            TabIndex
                         = 20
            ToolTipText
                          = "Click here to print all variants."
                       = 2400
            Top
            Width
                        = 1215
35
          Begin VB.CommandButton cmdDisplayModel
            Caption
                        = "Display Model"
            Height
                       = 495
                      = -70680
            Left
40
            TabIndex
                         = 19
                          = "Click here to view the active model."
            ToolTipText
            Top
                       = 1320
            Width
                       = 1215
          End
```

Begin VB.ListBox lstDummy

```
Height
                       = 255
            ItemData
                        = "TCA.frx":0674
            Left
                      = 4680
                      = "TCA.frx":0676
            List
 5
                       = -1 'True
            Sorted
            TabIndex
                        = 18
                      = 7800
            Top
            Visible
                       = 0 'False
            Width
                       = 615
10
          End
          Begin VB.TextBox txtNum2Generate
            Height
                       = 315
                      = -74760
            Left
            TabIndex
                        = 16
            ToolTipText = "Enter the number variants to generate here."
15
            Top
                       = 1140
                       = 855
            Width
          End
          Begin VB.CommandButton cmdSetAttributes
            Caption
                        = "Set Attributes"
            Enabled
                        = 0 'False
            Height
                       = 495
            Left
                      = 4320
            TabIndex
                         = 15
25
            ToolTipText = "Click here to reset the attributes for this model family."
 IJ.
            Top
                       = 720
            Width
                       = 1215
          End
          Begin ComctlLib.TreeView treModels
                        = "TCA.frx":0678
            DragIcon
            Height
                       = 2955
                      = 240
            Left
            TabIndex
                        = 13
                          = "Left button click on a model to select it. Then right button click for
            ToolTipText
35
       options."
                      = 780
            Top
            Width
                       = 3855
            ExtentX
                        = 6800
            ExtentY
                        = 5212
                        = 327682
40
            Version
            LabelEdit
                        = 1
            LineStyle
                      = 7
            Style
            Appearance
                         = 1
```

```
Begin VB.ListBox lstConstraints
                        = "TCA.frx":07C2
           DragIcon
            Height
                       = 1635
            Index
                       = 1
                        = "TCA.frx":0ACC
 5
            ItemData
            Left
                      = -74760
                      = "TCA.frx":0ACE
            List
                      = 1 'Checkbox
            Style
                        = 10
            TabIndex
10
            ToolTipText
                          = "Left button click to select a constraint. Then right button click for
       constraint options."
            Top
                      = 5520
            Width
                       = 3855
          End
          Begin VB.CommandButton cmdTestAll
15
            Caption
                        = "Test All"
            Height
                       = 495
                      = -70680
            Left
                        = 8
            TabIndex
            ToolTipText = "Click here to test all checked variables and constraints."
                       = 1320
            Top
 Ш
            Width
                       = 1215
          End
 Begin VB.CommandButton cmdSaveModel
25=
                       = "Save Model"
            Caption
 41
            Height
                       = 495
            Left
                      = -70680
            TabIndex
                        = 7
            ToolTipText = "Click here to save this model."
            Top
                       = 720
            Width
                       = 1215
          Begin VB.CommandButton cmdImportConstraints
            Caption
                        = "Import Constraints"
35
            Height
                       = 495
                      = -70680
            Left
                        = 6
            TabIndex
            ToolTipText
                          = "Click here to import a variable/constraint set."
            Top
                       = 1920
            Width
                       = 1215
40
          Begin VB.CommandButton cmdExportConstraints
                       = "Export Constraints"
            Caption
            Height
                       = 495
                      = -70680
45
            Left
```

```
TabIndex
            ToolTipText = "Click here to export a variable/constraint set."
            Top
                       = 2520
            Width
                       = 1215
 5
           End
           Begin VB.CommandButton cmdGenerate
                        = "Generate"
            Caption
            Height
                       = 495
            Left
                      = -70680
10
                         = 4
            TabIndex
            ToolTipText = "Click here to generate variants."
            Top
                       = 720
            Width
                       = 1215
          End
15
          Begin VB.TextBox txtVariablize
            BackColor
                         = &H8000000C&
            Height
                       = 375
            Left
                      = 5880
            TabIndex
                        = 2
                      = "Rob"
            Text
                      = 4740
            Top
25
            Visible
                       = 0 'False
            Width
                       = 615
          End
          Begin VB.ListBox lstAccepted
 4ĵ
            Height
                       = 2985
            ItemData
                        = "TCA.frx":0AD0
            Left
                      = 240
            List
                      = "TCA.frx":0AD2
            MultiSelect = 2 'Extended
            TabIndex
                        = 55
            ToolTipText = "Left button click on a variant to view it. Then right button click for
       options."
            Top
                      = 4320
35
            Width
                       = 3855
          End
          Begin VB.Label lblAccepted
            Caption
                       = "Accepted variants"
            Height
                       = 255
40 .
            Left
                      = 240
            TabIndex
                        = 32
                      = 4080
            Top
            Width
                       = 2535
          End
45
          Begin VB.Label lblDiff
```

```
= "Prolog randomization:"
            Caption
           Height
                       = 255
            Left
                     = -73440
            TabIndex
                        = 28
 5
            Top
                      = 840
            Width
                       = 1935
          End
          Begin VB.Label Label1
            Caption
                       = "High"
10
           Height
                       = 255
           Left
                     = -71760
           TabIndex
                        = 27
            Top
                      = 1440
            Width
                       = 495
15
          End
          Begin VB.Label lblMed
                       = "Medium"
           Caption
           Height
                      = 255
           Left
                     = -72720
           TabIndex
                        = 26
                      = 1440
           Top
 Width
                      = 735
          End
          Begin VB.Label lblLow
25<u>.</u>
           Caption
                       = "Low"
                      = 255
           Height
           Left
                     = -73440
 C.
           TabIndex
                        = 25
           Top
                      = 1440
30
           Width
                      = 495
          End
          Begin VB.Label lblDummy
           BorderStyle = 1 'Fixed Single
           Height
                      = 375
35
           Left
                     = 4680
           TabIndex
                        = 23
           Top
                      = 6840
           Visible
                      = 0 'False
           Width
                      = 615
40
          End
          Begin VB.Label lblVariants
           Caption
                       = "Variants"
           Height
                      = 255
                     = -74760
           Left
```

TabIndex

= 22

```
Top
                      = 2280
           Width
                      = 2055
          Begin ComctlLib.ImageList imlI
5
           Left
                     = 4680
           Top
                     = 7200
            ExtentX
                        = 1005
            ExtentY
                        = 1005
           BackColor
                        = -2147483643
           ImageWidth
10
                         = 16
           ImageHeight
                         = 16
                       = 12632256
           MaskColor
            Version
                       = 327682
           BeginProperty Images {0713E8C2-850A-101B-AFC0-4210102A8DA7}
15
             NumListImages = 2
             BeginProperty ListImage1 {0713E8C3-850A-101B-AFC0-4210102A8DA7}
                         = "TCA.frx":0AD4
              Picture
              Key
             EndProperty
             BeginProperty ListImage2 {0713E8C3-850A-101B-AFC0-4210102A8DA7}
                         = "TCA.frx":1026
              Picture
 <u>D</u>1
                         = ""
              Key
 IJ
 Frank Strait
             EndProperty
           EndProperty
25
          End
          Begin VB.Label LblNumVariants
                       = "Number:"
           Caption
           Height
                      = 255
           Left
                     = -74760
           TabIndex
                       = 17
                      = 900
           Top
           Width
                      = 735
          End
          Begin VB.Label lblFamily
35
           Caption
                       = "Family members"
                      = 255
           Height
           Left
                     = 240
           TabIndex
                      = 14
                      = 540
           Top
           Width
                      = 3615
40
          End
          Begin VB.Label lblDistractor
                       = "Distractor Constraints"
           Caption
           Height
                      = 255
                     = -74760
45
           Left
```

```
TabIndex
                        = 12
                      = 5280
           Top
           Width
                      = 2535
          End
 5
          Begin VB.Label lblCloningConstraints
           Caption
                       = "Variation Constraints"
                       = "TCA.frx":1578.
           DragIcon
           Height
                      = 255
           Left
                     = -74760
                        = 11
10
           TabIndex
                     = 2880
           Top
           Width
                      = 2535
          End
          Begin VB.Label lblVariables
15
           Caption
                       = "Variables"
           Height
                      = 255
           Left
                     = -74760
                        = 9
           TabIndex
           Top
                     = 480
           Width
                      = 855
          End
 đ١
 Ū
        End
 Begin ComctlLib.StatusBar stbS
          Align
                    = 2 'Align Bottom
25
          Height
                     = 300
 Ţ,
          Left
                    = 0
          TabIndex
                      = 30
 C.
          Top
                    = 8010
          Width
                     = 11400
          ExtentX
                      = 20108
          ExtentY
                      = 529
                      = ""
          SimpleText
 C
           Version
                      = 327682
          BeginProperty Panels {0713E89E-850A-101B-AFC0-4210102A8DA7}
           NumPanels
35
                         = 11
           BeginProperty Panel1 {0713E89F-850A-101B-AFC0-4210102A8DA7}
             Alignment
                          = 2
             AutoSize
                         = 2
             Bevel
                        = 0
             Object.Width
40
                              = 2117
             MinWidth
                          = 2117
                       = "Program:"
             Text
                         = "Program:"
             TextSave
             Key
             Object.Tag
45
```

```
EndProperty
           BeginProperty Panel2 {0713E89F-850A-101B-AFC0-4210102A8DA7}
                         = 1
             Alignment
                        = 2
             AutoSize
             Object. Width
                             = 1058
 5
             MinWidth
                       = 1058
                      = ""
             Kev
             Object.Tag
           EndProperty
           BeginProperty Panel3 {0713E89F-850A-101B-AFC0-4210102A8DA7}
10
             Alignment
                         = 2
             AutoSize
                        = 2
                       = 0
             Bevel
             Object. Width
                           = 1773
             MinWidth
                         = 1764
15
             Text
                      = "Family:"
                         = "Family:"
             TextSave
                       = ""
             Key
                            = ""
             Object.Tag
           EndProperty
           BeginProperty Panel4 {0713E89F-850A-101B-AFC0-4210102A8DA7}
 Øì
             Alignment
                         = 1
 Ų٦
                        = 2
 Marin allen
             AutoSize
                             = 2646
             Object. Width
25
             MinWidth
                         = 2646
                       = ""
 넯
             Key
             Object.Tag
           EndProperty
           BeginProperty Panel5 {0713E89F-850A-101B-AFC0-4210102A8DA7}
             Alignment
                         = 2
             AutoSize
                        = 2
                       = 0
             Bevel
             Object. Width
                             = 2117
             MinWidth
                         = 2117
35
             Text
                      = "Attributes:"
             TextSave
                         = "Attributes:"
            Key
             Object.Tag
           EndProperty
           BeginProperty Panel6 {0713E89F-850A-101B-AFC0-4210102A8DA7}
40
             Alignment
                         = 1
                        = 2
             AutoSize
             Object.Width
                             = 1058
             MinWidth
                         = 1058
             Key
```

```
Object.Tag
           EndProperty
           BeginProperty Panel7 {0713E89F-850A-101B-AFC0-4210102A8DA7}
             Alignment
                         = 2
             AutoSize
 5
             Object.Width
                              = 1058
             MinWidth
                          = 1058
                       = ""
             Key
             Object.Tag
10
           EndProperty
           BeginProperty Panel8 {0713E89F-850A-101B-AFC0-4210102A8DA7}
             AutoSize
             Object. Width
                              = 1058
             MinWidth
                          = 1058
15
             Key
             Object.Tag
           EndProperty
           BeginProperty Panel9 {0713E89F-850A-101B-AFC0-4210102A8DA7}
             Alignment
             AutoSize
                         = 2
                        = 0
             Bevel
             Object.Width
                              = 2487
 MinWidth
                          = 2469
                       = "Active Model:"
             Text
25
                         = "Active Model:"
             TextSave
 ű
             Key
             Object.Tag
           EndProperty
           BeginProperty Panel10 {0713E89F-850A-101B-AFC0-4210102A8DA7}
             Alignment
                         = 2
             AutoSize
             Object.Width
                              = 450
             MinWidth
                          = 441
                         1111
             Key
             Object.Tag
35
            EndProperty
            BeginProperty Panel11 {0713E89F-850A-101B-AFC0-4210102A8DA7}
             Alignment
                          = 1
                         = 2
             AutoSize
                              = 2646
             Object.Width
40
                          = 2646
             MinWidth
             Key
             Object.Tag
            EndProperty
          EndProperty
```

	End
	Begin VB.Menu mnuFile
	Caption = "File"
	Begin VB.Menu mnuFileNew
5	Caption = "New"
	End
	Begin VB.Menu mnuFileOpen
	Caption = "Open"
	End
10	Begin VB.Menu mnuFileImportItem
	Caption = "Import Locked Item"
	End
	Begin VB.Menu mnuFileSaveAs
	Caption = "Save As"
15	Visible = 0 'False
10	End
	Begin VB.Menu mnuFileSave
	Caption = "Save"
	Visible = 0 'False
20	End
2017	Begin VB.Menu mnuFilePrintSetup
<u>1</u> 33	Caption = "Print Setup"
	End
199 169	Begin VB.Menu mnuFileExit
25 🖫	Caption = "Exit"
20g	End
, R	End
[]	Begin VB.Menu mnuHelp
4î	Caption = "Help"
30=]	NegotiatePosition= 3 'Right
30 _F	Begin VB.Menu mnuHelpAbout
	Caption = "About"
	End
	End
35	Begin VB.Menu mnuVariables
33	Caption = "Variables"
	Visible = 0 'False
	Begin VB.Menu mnuVariablesAdd
	Caption = "Add"
40	End
40	
	Begin VB.Menu mnuVariablesEdit
	Caption = "Edit"
	End
4.5	Begin VB.Menu mnuVariablesRemove
45	Caption = "Remove"

	End
	Begin VB.Menu mnuVariablesRemoveAll
	Caption = "Remove All"
	End
5	Begin VB.Menu mnuVariablesEnableAll
	Caption = "Enable All"
	End
	Begin VB.Menu mnuVariablesDisableAll
	Caption = "Disable All"
10	End
	Begin VB.Menu mnuVariablesTest
	Caption = "Test"
	End
	End
15	Begin VB.Menu mnuConstraints
	Caption = "Constraints"
	Visible $= 0$ 'False
	Begin VB.Menu mnuConstraintsAdd
	Caption = "Add"
20 [End
T1	Begin VB.Menu mnuConstraintsEdit
Ū1	Caption = "Edit"
rá te Pa	End
1	Begin VB.Menu mnuConstraintsRemove
25	Caption = "Remove"
	End
= ====	Begin VB.Menu mnuConstraintsRemoveAll
C) 4)	Caption = "Remove All"
en en	End
30≒⁴	Begin VB.Menu mnuConstraintsEnableAll
- 1	Caption = "Enable All"
_ 1	End
	Begin VB.Menu mnuConstraintsDisableAll
2.5	Caption = "Disable All"
35	End
	Begin VB.Menu mnuConstraintsTest Caption = "Test"
	Cuption
	End
40	End
40	Begin VB.Menu mnuDisp
	Caption = "Disposition" Visible = 0 'False
	7 101010
	Begin VB.Menu mnuDispAccept Caption = "Accept"
45	Caption = "Accept" End
43	Liid

odel
;
file
ру
1 3
ste
alse

'contains family

```
' word
        Private mudtWord As MSWord
        ' prolog activex
        Private mudtProlog As Prolog
 5
        ' needed for SetAllCheckboxes sub
        Private mlstCurrentListBox As ListBox
        ' needed so frmConstraint know which kind of constraint to create
        Private mintConstrLBInd As Integer
        ' used as a flag when mnuFileImportLockedItem calls mnuFileNew
10
        Private mudtItemType As ItemType
        'holding area for copy / paste of variant profiles
        Private mudtClone As Clone
 C)
 J.
        ' turn full window drag back on if this is TRUE
 đ١
15
        Private mblnRestoreFullWindowDrag As Boolean
 Men alben Chin alben
Sant n n Sant u n
        Public Enum EditFlags
          aeNothing = 0
          aeAdd = 1
          aeEdit = 2
        End Enum
        Public Enum TestType
          tcTestVariables = 0
          tcTestVariationConstraints = 1
          tcTestDistractorConstraints = 2
          tcTestAll = 4
25
        End Enum
        ' for importing/exporting variables and constraints
        Private Enum ConstraintRecordLayout
          crVariableIndex = 1 ' 4 byte long
          crConstraintIndex = 5 ' 4 byte long
30
          crVariables = 9 'binary - variable size
        End Enum
        Private Enum IconImage
```

Private mudtFam As Family

imSnowflake = 1

```
imSun = 2
        End Enum
       ' used to update status bar
       Private Enum PanelIndex
 5
          pnProgramCaption = 1
          pnProgramName = 2
          pnFamilyCaption = 3
          pnFamilyName = 4
          pnAttributesCaption = 5
          pnItemType = 6
10
          pnGeneric = 7
          pnProximity = 8
          pnActiveModelCaption = 9
          pnActiveModelIcon = 10
          pnActiveModelName = 11
15
       End Enum
       Public Property Get Family() As Family
 IJ.
          Set Family = mudtFam
 Q1
Li
Man abai
       End Property
       Public Property Let Family(ByVal udtNewValue As Family)
20
 Ų,
          mudtFam = udtNewValue
 T)
       End Property
 <u>ا</u>ة
       Private Sub cmdCancel Click()
       End Sub
       Private Sub cmdAcceptedCopy_Click()
25
          Call mnuAcceptedCopy Click
       End Sub
       Private Sub cmdAcceptedEdit_Click()
          Call mnuAcceptedProfile Click
30
       End Sub
```

	Private Sub cmdAcceptedPaste_Click()
	Call mnuAcceptedPaste_Click
	End Sub
	Private Sub cmdComments_Click()
5	frmComments.Comment = mudtFam.ActiveModel.Comments frmComments.Show vbModal mudtFam.ActiveModel.Comments = frmComments.Comment
	UpdateTab1ControlStates
10	End Sub
	Private Sub cmdConstraintAdd_Click(index As Integer)
	mintConstrLBInd = index Call mnuConstraintsAdd_Click
15 <u>4</u>	End Sub
हिंदी, स्ट्रीस्त हिंदी, स्ट्रीस सक्ती १६ १ १ व्याप्त ११ १	Private Sub cmdConstraintEdit_Click(index As Integer)
E	mintConstrLBInd = index Call mnuConstraintsEdit_Click
T 67 65 7	End Sub
20=1	Private Sub cmdConstraintRemove_Click(index As Integer)
E)	mintConstrLBInd = index Call mnuConstraintsRemove_Click
	End Sub
	Private Sub cmdConstraintTest_Click(index As Integer)
25	mintConstrLBInd = index Call mnuConstraintsTest_Click
	End Sub
	Private Sub cmdDispAccept_Click()

```
End Sub
       Private Sub cmdDispDefer_Click()
          Call mnuDispDefer Click
 5
       End Sub
       Private Sub cmdDispDiscard Click()
          Call mnuDispDiscard Click
       End Sub
       Private Sub cmdDisplayModel Click()
          Call mudtFam.ActiveModel.OpenDoc(mudtWord)
       End Sub
 ۵ì
 We also the desiration than
       Private Sub cmdDispMakeModel_Click()
          Call mnuDispMakeModel Click
End Sub
       Private Sub cmdDone Click()
          Dim intI As Integer
          Dim udtClone As Clone
          Dim dMode As String
          Dim iType As String
20
          Dim key As String
          Dim Program As String
          Dim root As String
          Dim udtFamIni As New IniFile
          Dim udtProgress As New Progress
         If MsgBox("Prepare this family for export to TCS?", _
25
            vbOuestion + vbYesNo) = vbNo Then
            Exit Sub
```

End If

Call mnuDispAccept Click

```
If mudtFam.ActiveModel Is Nothing Then
            ' do nothing
          Else
            mudtFam.ActiveModel.WriteModel
 5
          End If
          ' close this so it can be copied to the out directory
          mudtFam.ActiveModel.CloseDoc
          Call udtProgress.Init(mudtFam.Clones.Count + 2, "Preparing family for exporting to TCS...")
          udtProgress.Advance
10
          root = ExtractFileNameNoExt(mudtFam.FileName)
          udtFamIni.FN = OUT DIRECTORY & root & ".ini"
          Select Case mudtFam.Program
            Case prGRE
               Program = "GRE"
            Case prGMAT
               Program = "GMAT"
Her abou His abou Han A.
            Case prSAT
               Program = "SAT"
            Case prMR
               Program = "MR"
          End Select
          Dim udtInIni As New IniFile
          udtInIni.FN = left(mudtFam.FileName, Len(mudtFam.FileName) - 3) & ____
            "ini"
          Dim strModelNo As String
25
          ' started with a locked item (during this session)
          strModelNo = udtInIni.GetProfileString("LockedItemData", _
            "TCAModelNo")
          ' started with an existing family (during this session)
          If strModelNo = "Not Found" Then
30
            strModelNo = udtInIni.GetProfileString("Family", _
               "TCAModelNo")
          End If
```

```
Call udtFamIni.SetKeyValuePair("LockedAccnum", mudtFam.AccNum)
         Call udtFamIni.SetKeyValuePair("Program", Program)
         Dim strProx As String
 5
         Select Case mudtFam.Proximity
            Case prNear
              strProx = "close"
            Case prMedium
              strProx = "medium"
10
            Case prFar
              strProx = "far"
         End Select
         Call udtFamIni.SetKeyValuePair("Proximity", strProx)
         If mudtFam.Generic Then
15
            Call udtFamIni.SetKeyValuePair("Nature", "generic")
            Call udtFamIni.SetKeyValuePair("Nature", "non-generic")
         End If
         For Each udtClone In mudtFam.Clones
20 ፟
           udtClone.CloseDoc
           If udtClone.IsRouted = False Then
              dMode = "TCA"
              iType = "TCA"
 ļ.
 C)
              Call FileCopy(IN DIRECTORY & udtClone.FileName,
                     OUT DIRECTORY & udtClone.FileName)
25
           Else
              If udtClone.DeliveryMode = dmPPT Then
               dMode = "PPT"
              Else
30
               dMode = "CBT"
              End If
              Call udtClone.OpenDoc(mudtWord, IN DIRECTORY)
              Select Case mudtFam.ItemType
```

Call udtFamIni.SetKeyValuePair("TCAModelNo", strModelNo)

```
Case ptStandardMC
                  If dMode = "PPT" Then
                    iType = "MC Item"
                    Call genPPT MultChoice(udtClone, key)
 5
                  Else
                    iType = "QANTDISC"
                    Call genCBT MultChoice(udtClone, key)
                  End If
                Case ptQuantComp
                  If dMode = "PPT" Then
10
                    iType = "QC Discrete"
                    Call genPPT QuantComp(udtClone, key)
                  Else
                    iType = "QANTCOMP"
15
                    Call genCBT QuantComp(udtClone, key)
                  End If
                Case ptDataSuff
                  iType = "DATASUFF"
 ű,
                  Call genCBT_DataSuff(udtClone, key)
 đì
 Li
20=
              End Select
              udtClone.CloneDoc.Close
            End If
 43
            Dim udtClnIni As New IniFile
 lii k
            root = ExtractFileNameNoExt(udtClone.FileName)
            Call udtFamIni.SetKeyValuePair("Variant", root)
253
            udtClnIni.FN = OUT DIRECTORY & root & ".ini"
            Call udtClnIni.SetKeyValuePair("DeliveryMode", dMode)
            Call udtClnIni.SetKeyValuePair("Key", udtClone.key)
            Call udtClnIni.SetKeyValuePair("ItemType", iType)
            Call udtClnIni.WriteProfileSection("Variant")
30
            Call udtClnIni.WriteProfileString("Exit", " ", " ")
            Set udtClnIni = Nothing
            udtProgress.Advance
```

Next udtClone

```
'delete profiled variants from lstAccepted
          With lstAccepted
            intI = .ListCount - 1
 5
            Do While intI > -1
              Set udtClone = mudtFam.Clones.Item(Str(.ItemData(intI)))
              If udtClone.IsRouted Then
                ' remove the clone from the collection
                Call mudtFam.Clones.Remove(Str(.ItemData(intI)))
                ' remove it from the list box
10
                Call .RemoveItem(intI)
              End If
              intI = intI - 1
            Loop
15
          End With
         mudtFam.WriteFamily
 Dim fName As String
 4
         Dim strWildCard As String
 Ø1
 Uī
         For intI = 1 To treModels.Nodes.Count
            root = ExtractFileNameNoExt(treModels.Nodes.Item(intI))
20
            fName = root & ".doc"
            Call udtFamIni.SetKeyValuePair("Member", fName)
            Call FileCopy(IN_DIRECTORY & fName, OUT DIRECTORY & fName)
            fName = root & ".mdl"
            Call udtFamIni.SetKeyValuePair("Member", fName)
            Call FileCopy(IN_DIRECTORY & fName, OUT DIRECTORY & fName)
            If intI = 1 Then
              fName = root & ".mdf"
              strWildCard = root & "*.*"
              Call udtFamIni.SetKeyValuePair("Member", fName)
30
              Call FileCopy(IN DIRECTORY & fName, OUT DIRECTORY & fName)
            End If.
         Next
         Call udtFamIni.WriteProfileSection("Family")
         Call udtFamIni.WriteProfileString("Exit", " ", " ")
35
```

	mudtWord.WordApp.Documents.Close
5	Kill IN_DIRECTORY & strWildCard
	If strModelNo <> "Not Found" Then Kill IN_DIRECTORY & strModelNo & ".*" End If
	udtProgress.Advance
10	UpdateTab0ControlStates
	End Sub
	Private Sub genPPT_MultChoice(udtClone As Clone, itmKey As String)
ene ene en	Dim docTCAModel As Document Set docTCAModel = mudtWord.WordApp.Documents.Open(App.path & "\TCAClone.DOC")
-1 5 -	docTCAModel.Variables.Add "PROP_ACCNUM", "SSMCPPT"
	' mudtWord.WordApp.Run ("SetAccnum") mudtWord.WordApp.Run ("StartItem.Main")
20. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.	Dim tabchr As String tabchr = Chr(9) Dim destRange As Range Set destRange = docTCAModel.Content destRange.find.Style = "PPTStem" destRange.find.Execute FindText:=tabchr
	' MsgBox "PPT MultChoice"
25	udtClone.CloneDoc.Bookmarks("tca_Stem").Range.Copy destRange.Paste destRange.Borders.Enable = False destRange.ParagraphFormat.LeftIndent = InchesToPoints(0.25) destRange.Style = "PPTStem"
30	Dim respRange As Range Dim abcde As String abcde = "ABCDE"
	VBSCA -143-

mudtWord.WordApp.Documents.Open FileName:=App.path & "\tcaclone.doc"

ClearControls

```
For i = 1 To 5
           Set respRange = udtClone.CloneDoc.Bookmarks("tca Resp" & Mid(abcde, i, 1)).Range
           respRange.start = respRange.start + 4
           respRange.Copy
 5
           Set destRange = docTCAModel.Content
           destRange.find.Style = "PPTOptions"
           destRange.find.Execute FindText:="(" & Mid(abcde, i, 1) & ")"
           destRange.start = destRange.start + 4
10
           destRange.Paste
           destRange.Borders.Enable = False
           destRange.ParagraphFormat.LeftIndent = InchesToPoints(0.25)
           destRange.Style = "PPTOptions"
         Next
         Dim key As String
         key = udtClone.CloneDoc.Bookmarks("tca Key").Range.Text
         key = Mid(key, 8, 1)
 Mer Men Men Sec.
         itmKey = key
         For i = 1 To 5
20
           If key = Mid(abcde, i, 1) Then
             key = Format(i)
             Exit For
           End If
         Next
         Dim keyRange As Range
         Dim keyStart As Long
          Set keyRange = docTCAModel.Content
         keyStart = keyRange.End - 1
          docTCAModel.Content.InsertAfter Text:=itmKey
         keyRange.SetRange start:=keyStart, End:=docTCAModel.Content.End
30
         docTCAModel.Bookmarks.Add Name:="prop Key", Range:=keyRange
          Dim tmpFName As String
          tmpFName = OUT_DIRECTORY & udtClone.FileName
          docTCAModel.Variables("PROP ACCNUM").Delete
          docTCAModel.Variables.Add "PROP ACCNUM", "TCAVARNT"
35
```

Dim i As Integer

docTCAModel.SaveAs tmpFName docTCAModel.Close

End Sub

10

25

30

Private Sub genCBT_MultChoice(udtClone As Clone, itmKey As String)

5 Dim docTCAModel As Document Set docTCAModel = mudtWord.WordApp.Documents.Open(App.path & "\TCAClone.DOC")

' MsgBox "CBT MultChoice"

docTCAModel.Variables.Add "PROP_ACCNUM", "SSMCCBT" mudtWord.WordApp.Run ("SetAccnum") mudtWord.WordApp.Run ("StartItem.Main")

Dim tabchr As String
tabchr = Chr(9)
Dim destRange As Range
Set destRange = docTCAModel.Content
destRange.find.Execute FindText:="Enter stem here."

udtClone.CloneDoc.Bookmarks("tca_Stem").Range.Copy destRange.Paste destRange.Borders.Enable = False

Dim respRange As Range Dim abcde As String abcde = "ABCDE" Dim i As Integer

Set destRange = docTCAModel.Content destRange.find.Execute FindText:="Enter responses here" destRange.End = destRange.End + 1 destRange.Delete

For i = 1 To 5

Set respRange = udtClone.CloneDoc.Bookmarks("tca_Resp" & Mid(abcde, i, 1)).Range respRange.start = respRange.start + 4 respRange.Copy

destRange.Paste destRange.Style = "Choice" destRange.InsertParagraphAfter

Set destRange = destRange.Paragraphs(1).Next.Range

```
Next
         Dim key As String
         key = udtClone.CloneDoc.Bookmarks("tca Key").Range.Text
 5
         key = Mid(key, 8, 1)
         itmKey = key
         For i = 1 To 5
           If key = Mid(abcde, i, 1) Then
            key = Format(i)
10
            Exit For
           End If
         Next
         Dim keyRange As Range
         Dim keyStart As Long
         Set keyRange = docTCAModel.Content
15
         keyStart = keyRange.End - 1
 dry fire the first
         docTCAModel.Content.InsertAfter Text:=itmKey
         keyRange.SetRange start:=keyStart, End:=docTCAModel.Content.End
         docTCAModel.Bookmarks.Add Name:="prop Key", Range:=keyRange
20
         Dim tmpFName As String
         tmpFName = OUT DIRECTORY & udtClone.FileName
         docTCAModel.Variables("PROP ACCNUM").Delete
         docTCAModel.Variables.Add "PROP ACCNUM", "TCAVARNT"
 je j
         Call itemKey Store(docTCAModel, udtClone.key)
         docTCAModel.SaveAs tmpFName
25
         docTCAModel.Close
       End Sub
       Private Sub genPPT QuantComp(udtClone As Clone, itmKey As String)
         Dim docTCAModel As Document
         Set docTCAModel = mudtWord.WordApp.Documents.Open(App.path & "\TCAClone.DOC")
30
       ' MsgBox "PPT QuantComp"
         docTCAModel.Variables.Add "PROP ACCNUM", "QCPPT"
         mudtWord.WordApp.Run ("SetAccnum")
```

```
udtClone.CloneDoc.Bookmarks("tca Stem").Range.Copy
         docTCAModel.Tables(1).Cell(Row:=1, Column:=2).Range.Paste
         docTCAModel.Tables(1).Cell(Row:=1, Column:=2).Range.Style = "PPTQC StimCentered"
         udtClone.CloneDoc.Bookmarks("tca ColumnA").Range.Copy
 5
         docTCAModel.Tables(1).Cell(Row:=2, Column:=2).Range.Paste
         udtClone.CloneDoc.Bookmarks("tca ColumnB").Range.Copy
         docTCAModel.Tables(1).Cell(Row:=2, Column:=4).Range.Paste
         docTCAModel.Tables(1).Cell(Row:=2, Column:=2).Range.Style = "PPTQC AB"
         docTCAModel.Tables(1).Cell(Row:=2, Column:=4).Range.Style = "PPTQC AB"
10
         Dim key As String
         key = udtClone.CloneDoc.Bookmarks("tca Key").Range.Text
         key = Mid(key, 8, 1)
         itmKey = key
15
         Dim abcde As String
 Min alle fire also free
         abcde = "ABCDE"
         Dim i As Integer
         For i = 1 To 5
           If key = Mid(abcde, i, 1) Then
20
            kev = Format(i)
            Exit For
           End If
         Next
         Dim keyRange As Range
         Dim keyStart As Long
         Set keyRange = docTCAModel.Content
         keyStart = keyRange.End - 1
         docTCAModel.Content.InsertAfter Text:=itmKey
         keyRange.SetRange start:=keyStart, End:=docTCAModel.Content.End
         docTCAModel.Bookmarks.Add Name:="prop Key", Range:=keyRange
30
         Dim tmpFName As String
         tmpFName = OUT DIRECTORY & udtClone.FileName
         docTCAModel.Variables("PROP ACCNUM").Delete
         docTCAModel.Variables.Add "PROP_ACCNUM", "TCAVARNT"
```

VBSCA -147-

mudtWord.WordApp.Run ("StartItem.Main")

```
docTCAModel.Close
       End Sub
       Private Sub genCBT QuantComp(udtClone As Clone, itmKey As String)
 5
         Dim docTCAModel As Document
         Set docTCAModel = mudtWord.WordApp.Documents.Open(App.path & "\TCAClone.DOC")
       ' MsgBox "CBT QuantComp"
         docTCAModel.Variables.Add "PROP ACCNUM", "QCCBT"
         mudtWord.WordApp.Run ("SetAccnum")
10
         mudtWord.WordApp.Run ("StartItem.Main")
         udtClone.CloneDoc.Bookmarks("tca Stem").Range.Copy
         docTCAModel.Tables(1).Cell(Row:=1, Column:=1).Range.Paste
         udtClone.CloneDoc.Bookmarks("tca ColumnA").Range.Copy
         docTCAModel.Tables(1).Cell(Row:=2, Column:=1).Range.Paste
 <u>O</u>1
         udtClone.CloneDoc.Bookmarks("tca ColumnB").Range.Copy
 Mr. Jr. Mr.
         docTCAModel.Tables(1).Cell(Row:=2, Column:=2).Range.Paste
         Dim key As String
         key = udtClone.CloneDoc.Bookmarks("tca Key").Range.Text
         key = Mid(key, 8, 1)
         itmKey = key
         Dim abcde As String
         abcde = "ABCDE"
         Dim i As Integer
         For i = 1 To 5
25
           If key = Mid(abcde, i, 1) Then
            key = Format(i)
             Exit For
           End If
         Next
30
         Dim keyRange As Range
         Dim keyStart As Long
         Set keyRange = docTCAModel.Content
         keyStart = keyRange.End - 1
```

docTCAModel.SaveAs tmpFName

Set srcRange = udtClone.CloneDoc.Bookmarks("tca fStatement").Range

VBSCA -149-

docTCAModel.Content.InsertAfter Text:=itmKey

keyRange.SetRange start:=keyStart, End:=docTCAModel.Content.End docTCAModel.Bookmarks.Add Name:="prop Key", Range:=keyRange

```
srcRange.End = srcRange.End - 1
          If Len(srcRange.Text) > 0 Then
            srcRange.Copy
            destRange.Paste
 5
          End If
          destRange.Collapse Direction:=wdCollapseEnd
          destRange.InsertParagraphAfter
          destRange.Collapse Direction:=wdCollapseEnd
          Set srcRange = udtClone.CloneDoc.Bookmarks("tca sStatement").Range
10
          srcRange.End = srcRange.End - 1
          If Len(srcRange.Text) > 0 Then
            srcRange.Copy
            destRange.Paste
          End If
15
          Dim n As Integer
          n = docTCAModel.ListParagraphs.Count
          While n > 2
 The second
            Set destRange = docTCAModel.ListParagraphs(n).Range
            destRange.Delete
            n = n - 1
          Wend
          Dim key As String
          key = udtClone.CloneDoc.Bookmarks("tca_Key").Range.Text
          key = Mid(key, 8, 1)
25
          itmKey = key
 Ľ)
          Dim abcde As String
 片
          abcde = "ABCDE"
          Dim i As Integer
          For i = 1 To 5
30
           If key = Mid(abcde, i, 1) Then
             key = Format(i)
             Exit For
           End If
          Next
35
          Dim keyRange As Range
          Dim keyStart As Long
          Set keyRange = docTCAModel.Content
          keyStart = keyRange.End - 1
```

```
docTCAModel.Content.InsertAfter Text:=itmKey
         keyRange.SetRange start:=keyStart, End:=docTCAModel.Content.End
         docTCAModel.Bookmarks.Add Name:="prop Key", Range:=keyRange
         Dim tmpFName As String
 5
          tmpFName = OUT DIRECTORY & udtClone.FileName
          docTCAModel.Variables("PROP_ACCNUM").Delete
          docTCAModel. Variables. Add "PROP ACCNUM", "TCAVARNT"
          Call itemKey Store(docTCAModel, udtClone.key)
          docTCAModel.SaveAs tmpFName
          docTCAModel.Close
10
       End Sub
       Private Sub itemKey_Store(doc As Document, ByVal key As String)
         Dim i As Integer
         For i = 1 To 5
           If key = Mid("ABCDE", i, 1) Then
             key = Format(i)
 May affer May Act him
             Exit For
           End If
         Next
20
C1
         doc. Variables. Add "ItemKeyStore", key
       End Sub
       Private Sub cmdPrintConstraints Click()
         Dim udtV As Variable
         Dim udtC As Constraint
25
         Dim udtVI As VarInteger
         Dim udtVR As VarReal
         Dim udtVF As VarFraction
         Dim udtVS As VarString
         Dim udtP As New PrintModel
         Dim varS As Variant
30
         Dim varS1 As Variant
         Dim udtSS As SubString
         Dim intI As Integer
35
         udtP.ModelName = ExtractFileNameNoExt(mudtFam.ActiveModel.FileName)
```

For Each udtV In mudtFam.ActiveModel.Variables Call udtP.PrintString("Variable name: " & udtV.name, 2) 5 Select Case udtV.Typ Case vtInteger Call udtP.PrintString("Type: Integer", 3) Case vtReal Call udtP.PrintString("Type: Real", 3) 10 Case vtFraction Call udtP.PrintString("Type: Fraction", 3) Case vtString Call udtP.PrintString("Type: String", 3) 15 Case vtUntyped Call udtP.PrintString("Type: Untyped", 3) End Select If udtV.Enabled Then Call udtP.PrintString("Status: Enabled", 3) Else Call udtP.PrintString("Status: Disabled", 3) End If If udtV.Checksum Then Call udtP.PrintString("Checksum: Enabled", 3) Else Call udtP.PrintString("Checksum: Disabled", 3) End If Select Case udtV.Typ Case vtInteger Set udtVI = udtVIf udtVI.IsIndependent Then Call udtP.PrintString("Is independent = True," & _ " Range: from " & udtVI.From & " to " & udtVI.Too & " by " & udtVI.By, 3) 35 Else Call udtP.PrintString("Is independent = False", 3) End If Case vtReal Set udtVR = udtV40 If udtVR.IsIndependent Then Call udtP.PrintString("Is independent = True," & "Range: from " & udtVR.From & " to " & udtVR.Too & " by " & udtVR.By, 3) 45

Call udtP.PrintString("Variables:", 1)

	Else
•	Call udtP.PrintString("Is independent = False", 3)
	End If
	If udtVR.IsOnGrid Then
5	Call udtP.PrintString("Force on grid value: True", 3)
-	Else
	Call udtP.PrintString("Force on grid value: False", 3)
	End If
	Call udtP.PrintString("# Decimal places: " & _
10	Str(udtVR.DecimalPlaces), 3)
	If udtVR.TrailingZeros Then
	Call udtP.PrintString("Display trailing zeros: True", 3)
	Else
	Call udtP.PrintString("Display trailing zeros: False", 3)
15	End If
	Case vtFraction
	Set $udtVF = udtV$
	If udtVF.IsIndependent Then
₽5	Call udtP.PrintString("Is independent = True," & _
20	" Range: from " & udtVF.FromNumerator & _
200	"/" & udtVF.FromDenominator & _
	" to " & udtVF.ToNumerator & _
	"/" & udtVF.ToDenominator & _
157	" by " & udtVF.ByNumerator & _
25	"/" & udtVF.ByDenominator, 3)
	Else
e mag	Call udtP.PrintString("Is independent = False", 3)
<u>L</u>]	End If
	If udtVF.MixedNumbers Then
3 0	Call udtP.PrintString("Display mixed number: True", 3)
. C 1	Else
	Call udtP.PrintString("Display mixed number: False", 3
	End If
	Case vtString
35	Set $udtVS = udtV$
	If udtVS.IsIndexed Then
	Call udtP.PrintString("Indexed: True", 3)
	Call udtP.PrintString("Value Sets:", 3)
40	For Each varS In udtVS.StringCollection
40	Set udtSS = New SubString
	udtSS.Delimiter = Chr(STRING_DELIMITER)
	udtSS.StringValue = varS
	Call udtP.PrintString("Values:", 4)
45	intI = 1 For Each varS1 In udtSS.StringCollection
4 J	FOI Each vars) in adds.SunigConcolon

```
Call udtP.PrintString(Str(intI) & ". " & varS1, 5)
                         intI = intI + 1
                      Next varS1
                    Next varS
 5
                 Else
                    Call udtP.PrintString("Indexed: False", 3)
                    Call udtP.PrintString("Values:", 3)
                    For Each varS In udtVS.StringCollection
                       Call udtP.PrintString(varS, 4)
10
                    Next varS
                  End If
               Case vtUntyped
            End Select
15
          Next udtV
          Call udtP.PrintString("Constraints:", 1)
          Call udtP.PrintString("Variation constraints:", 2)
          For Each udtC In mudtFam.ActiveModel.Constraints
25
25
            If udtC.ConstraintType = ctVariation Then
               Call udtP.PrintString("Constraint: " & udtC.ConstraintString, 3)
               If udtC.Enabled Then
                  Call udtP.PrintString("Status: Enabled", 4)
               Else
                  Call udtP.PrintString("Status: Disabled", 4)
               End If
             End If
          Next udtC
          'exit if not MC
          If Not mudtFam.ItemType = ptStandardMC Then Exit Sub
35
          Call udtP.PrintString("Distractor constraints:", 2)
          For Each udtC In mudtFam.ActiveModel.Constraints
             If udtC.ConstraintType = ctDistractor Then
               Call udtP.PrintString("Constraint: " & udtC.ConstraintString, 3)
               If udtC.Enabled Then
40
                  Call udtP.PrintString("Status: Enabled", 4)
                  Call udtP.PrintString("Status: Disabled", 4)
               End If
45
             End If
```

	End Sub
	Private Sub cmdSetAttributes_Click()
5	frmAttributes.Show vbModal
10	If frmAttributes.OK Then mudtFam.Generic = frmAttributes.Generic mudtFam.Proximity = frmAttributes.Proximity mudtFam.IsDirty = True 'save family mudtFam.WriteFamily UpdateFamilyAttributes End If
	End Sub
15	Private Sub cmdTreeExtend_Click()
	Call mnuTreeExtend_Click
40 111 11	End Sub
	Private Sub cmdTreeRemove_Click()
1 20 2	Call mnuTreeRemove_Click
20	End Sub
	Private Sub cmdVariableAdd_Click()
	Call mnuVariablesAdd_Click frmVariable.Model = mudtFam.ActiveModel frmVariable.ListBox = lstVariables
25	frmVariable.Show vbModal
	UpdateTab1ControlStates
	End Sub
	Private Sub cmdVariableEdit_Click()

Next udtC

	Call mnuVariablesEdit_Click frmVariable.Model = mudtFam.ActiveModel frmVariable.ListBox = lstVariables
5	If lstVariables.ListIndex >= 0 Then ' Make sure list item is selected 'Set the key for access by frmVariable With lstVariables frmVariable.Variable =
10	mudtFam.ActiveModel.Variables.Item(Str(.ItemData(.ListIndex))) End With frmVariable.Show vbModal End If
	UpdateTab1ControlStates
	End Sub
	Private Sub cmdVariableRemove_Click()
1 5)	Call mnuVariablesRemove_Click
4	End Sub
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	Private Sub cmdVariableTest_Click()
	Call mnuVariablesTest_Click
20. 20.	End Sub
$2\overline{0}$	Private Sub Form_Initialize()
	frmSplash.Show
	End Sub
	Private Sub Form_Load()
25	' to trap cancels cdlCD.CancelError = True
	'Create Word Object Set mudtWord = New MSWord
30	' get rid of the kill file if it exists, as it will prevent ' StartProlog from working

'Create the Prolog object If mudtProlog Is Nothing Then Set mudtProlog = CreateObject("AXProlog.Prolog") 5 If Not mudtProlog.StartProlog Then Call MsgBox("Prolog cannot be started.", vbExclamation, "Prolog error") End If End If 10 treModels.ImageList = imlI frmSplash.UnloadMe Me.Show UpdateTab0ControlStates 'copies ied files from a holding area, as TCS deletes them for ' reasons unknown. Call Kill("c:\tcs\working*.ied") Call FileCopy("c:\tcs\tcaied\dscbt.ied", "c:\tcs\working\dscbt.ied") Call Shell("attrib -r c:\tcs\working\dscbt.ied", vbHide) Call FileCopy("c:\tcs\tcaied\qccbt.ied", "c:\tcs\working\qccbt.ied") Call FileCopy("c:\tcs\tcaied\qcppt.ied", "c:\tcs\working\qcppt.ied") Call FileCopy("c:\tcs\tcaied\ssmccbt.ied", "c:\tcs\working\ssmccbt.ied") Call FileCopy("c:\tcs\tcaied\ssmcppt.ied", "c:\tcs\working\ssmcppt.ied") End Sub Private Sub Form MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single) Call sstMainTab MouseMove(Button, Shift, X, Y) End Sub Private Sub Form Resize() ' if minimized, don't resize 30 If Me. WindowState = vbMinimized Then Exit Sub Dim udtW As New Win32API Dim result As Long 'Turn off full window drag if it's on

DestroyKillFile

	If udtW.IsFullWindowDragOn Then udtW.TurnOffFullWindowDrag mblnRestoreFullWindowDrag = True End If
5	'adjust horizontals fraWord.left = 120 fraWord.Width = Me.Width - sstMainTab.Width - 360 sstMainTab.left = fraWord.Width + 180
10	'adjust verticals fraWord.Height = Me.Height - fraWord.top - stbS.Height - 700 ' approx title bar heigh sstMainTab.Height = fraWord.Height
	mudtWord.Resize
	End Sub
1 5	Private Sub Form_Unload(Cancel As Integer)
n de Kar (ku) de er b'n een met met de	'if no active family, hit the road If mudtFam Is Nothing Then 'do nothing
	Else mudtFam.WriteFamily If mudtFam.ActiveModel Is Nothing Then 'see if an active model has been set 'do nothing
251 251	Else mudtFam.ActiveModel.CloseDoc KillVariants 'Get rid of any variants left on tab 3 mudtFam.ActiveModel.WriteModel 'save the active model End If
	End If
30	' close all docs mudtWord.CloseAllDocs
	'Kill Word before frmTCA is unloaded to prevent automation error Set mudtWord = Nothing
35	' force event Call sstMainTab_MouseMove(1, 1, 1, 1)
33	' To cleanly shut down AXProlog on W95, 98 boxes mudtProlog.EndProlog

```
'End required by NT 4.0 to shut down TCA successfully!
          End
       End Sub
       Private Sub IstVariables ItemCheck(Item As Integer)
 5
          With lstVariables
            If lstVariables.ListCount = 0 Then Exit Sub 'this prevents an error
            If mudtFam.ActiveModel.IsFrozen Then
               .Selected(Item) =
                 mudtFam.ActiveModel.Variables.Item(Str(.ItemData(Item))).Enabled
10
            Else
               mudtFam.ActiveModel.Variables.Item(Str(.ItemData(Item))).Enabled =
                 .Selected(Item)
            End If
          End With
          UpdateTab1ControlStates
15
       End Sub
 <u>a</u>î
 Prin Bru Hin Ann Kin
       Private Sub lstVariables MouseDown(Button As Integer, Shift As Integer, __
          X As Single, Y As Single)
          Dim strIndex As String
          Set mlstCurrentListBox = lstVariables
          If Button = vbRightButton Then
            frmVariable.AddEditFlag = aeNothing
            PopupMenu mnuVariables 'Pull up popup menu for variable window
            frmVariable.Model = mudtFam.ActiveModel
            frmVariable.ListBox = lstVariables
25
            Select Case frmVariable.AddEditFlag
               Case aeEdit
                 If lstVariables.ListIndex >= 0 Then 'Make sure list item is selected
                    'Set the key for access by frmVariable
                    With lstVariables
30
                      frmVariable.Variable =
                        mudtFam.ActiveModel.Variables.Item(Str(.ItemData(.ListIndex)))
                   End With
                    frmVariable.Show vbModal
                 End If
35
               Case aeAdd
```

```
End Select
          End If
        End Sub
 5
        Private Sub lstConstraints ItemCheck(index As Integer, Item As Integer)
          Dim strKey As String
           With lstConstraints(index)
             If .ListCount = 0 Then Exit Sub 'prevents error if listbox is empty
             If mudtFam.ActiveModel.IsFrozen Then
                .Selected(Item) =
10
                  mudtFam.ActiveModel.Constraints.Item(Str(.ItemData(Item))).Enabled
             Else
               mudtFam.ActiveModel.Constraints.Item(Str(.ItemData(Item))).Enabled = _
                  .Selected(Item)
             End If
          End With
 The sheet the sheet time that
          UpdateTab1ControlStates
        End Sub
        'provide right button menu options
20
        Private Sub 1stConstraints MouseDown(index As Integer, Button As Integer,
          Shift As Integer, X As Single, Y As Single)
          Dim strIndex As String
          Set mlstCurrentListBox = lstConstraints(index)
          mintConstrLBInd = index
          Call UpdateTab1ControlStates(index)
25
          If Button = vbRightButton Then
             PopupMenu mnuConstraints
             If mudtFam.ActiveModel.IsFrozen = False Then
30
               lstConstraints(index).Drag
             End If
          End If
        End Sub
```

frmVariable.Show vbModal

```
'Enable drag and drop between constraint list boxes
        Private Sub 1stConstraints DragDrop(index As Integer, Source As Control,
          X As Single, Y As Single)
          If Source.ListCount = 0 Then
 5
             Exit Sub
          End If
          If index \diamond Source index Then 'Assure that it's another listbox!
             Dim udtConstraint As Constraint
             Dim strKey As String
             strKey = Str(Source.ItemData(Source.ListIndex))
10
             With lstConstraints(index)
               ' Add the dragged constraint to the end of the target listbox
               .List(.ListCount) = Source.List(Source.ListIndex)
               'Update the index in the new listbox entry
               .ItemData(.ListCount - 1) = Source.ItemData(Source.ListIndex)
             End With
 Wen offen Bin when they Kill
             'Find the constraint object being moved and update it's "type" in the collection
             Set udtConstraint = mudtFam.ActiveModel.Constraints.Item(strKey)
             udtConstraint.ConstraintType = index
             ' Delete the dragged constraint from the source listbox
             Call Source.RemoveItem(Source.ListIndex)
          End If
          UpdateTab1ControlStates
        End Sub
25
        Private Sub IstDisposition MouseDown(Button As Integer, Shift As Integer, _
          X As Single, Y As Single)
          Dim udtClone As Clone
          If Button = vbRightButton Then
             PopupMenu mnuDisp
30
          Else
             With lstDisposition
               If .ListCount > 0 Then 'a valid selection has been made
```

```
Set udtClone = mudtFam.ActiveModel.Clones.Item(Str(.ItemData(.ListIndex)))
                Call udtClone.OpenDoc(mudtWord, IN DIRECTORY)
              End If
            End With
 5
         End If
       End Sub
       Private Sub IstAccepted MouseDown(Button As Integer, Shift As Integer, X As Single, Y As
       Single)
          Static udtClone As Clone
10
         If Button = vbRightButton Then
            With lstAccepted
              If .SelCount = 1 Then
                 mnuAcceptedProfile.Enabled = True
                 mnuAcceptedCopy.Enabled = True
                 Set udtClone = mudtFam.Clones.Item(Str(.ItemData(.ListIndex)))
15
                 Call udtClone.OpenDoc(mudtWord, IN DIRECTORY)
                 Set udtClone = Nothing
 41
              Else
 Ωî
 U
                 mnuAcceptedProfile.Enabled = False
20.
                 mnuAcceptedCopy.Enabled = False
 ųĵ,
              End If
            End With
            PopupMenu mnuAccepted
          Else ' left button click
            If udtClone Is Nothing Then
              ' do nothing
            Else
              udtClone.CloseDoc
              Set udtClone = Nothing
30
            End If
            With lstAccepted
              If .ListCount > 0 Then
                Set udtClone = mudtFam.Clones.Item(Str(.ItemData(.ListIndex)))
                Call udtClone.OpenDoc(mudtWord, IN DIRECTORY)
35
              End If
            End With
         End If
         UpdateTab0ControlStates
       End Sub
```

	Private Sub cmdSaveModel_Click()
5	If mudtFam.ActiveModel.IsDirty Then mudtFam.ActiveModel.WriteModel KillVariants 'delete any variants on tab 3 End If
	UpdateTab1ControlStates
	End Sub
	Private Sub cmdTestAll_Click()
10	cmdSaveModel_Click ' force a save Call TestConstraints(tcTestAll)
	End Sub
	Private Sub cmdImportConstraints_Click()
	Dim strFN As String With cdlCD .FileName = "" .CancelError = True .DialogTitle = "Import constraints from file" .Filter = "Constraint Files (*.con) *.con " .DefaultExt = ".con" .InitDir = "c:\tcs\tca\constraints" .Flags = cdlOFNFileMustExist + cdlOFNHideReadOnly On Error GoTo Cancel ' trap the Cancel button .ShowOpen On Error GoTo 0 ' reset the error strFN = .FileName End With
	' exit if there's no file name
30	If Len(strFN) = 0 Then Exit Sub End If
	'create a new collection of imported variables
	Dim udtCVariables As New CVariables

' add the imported variables to the main collection Dim udtNewVar As Variable For Each udtNewVar In udtCVariables If mudtFam.ActiveModel.Variables.UniqueName(udtNewVar.name) Then 5 Call mudtFam.ActiveModel.Variables.AddObject(udtNewVar) With IstVariables ' Add the new variable to the variable list box Call .AddItem(udtNewVar.ScreenFormat) 'Set ItemData to index value of the variable object 10 .ItemData(.ListCount - 1) = udtNewVar.index ' Set the check box. .Selected(.ListCount - 1) = udtNewVar.Enabled End With Else 15 Call MsgBox("Variable " & udtNewVar.name & " will not be imported.", vbExclamation, "Variable not unique") End If Next udtNewVar ' read the imported constraints into a new collection Dim udtCConstraints As New CConstraints Call udtCConstraints.ReadCollection(strFN, crConstraintIndex, READ_UNTIL EOF) ' add the imported constraints Dim udtNewCon As Constraint 25 For Each udtNewCon In udtCConstraints If mudtFam.ActiveModel.Constraints.UniqueConstraint(udtNewCon.ConstraintString) Then Call mudtFam.ActiveModel.Constraints.AddObject(udtNewCon) With lstConstraints(udtNewCon.ConstraintType) ' Add the new variable to the variable list box 30 Call .AddItem(udtNewCon.ConstraintString) 'Set ItemData to index value of the variable object .ItemData(.ListCount - 1) = udtNewCon.index 'Check the check box .Selected(.ListCount - 1) = udtNewCon.Enabled 35

VBSCA -164-

Call udtCVariables.ReadCollection(strFN, crVariableIndex, crConstraintIndex)

```
End With
            Else
               Call MsgBox("Constraint " & udtNewCon.ConstraintString & " will not be imported.",
                 vbExclamation, "Constraint not unique")
 5
            End If
          Next udtNewCon
        Cancel:
          Exit Sub
        End Sub
        Private Sub cmdExportConstraints_Click()
10
          Dim strFN As String
          With cdlCD
            .FileName = ""
            .DialogTitle = "Export constraints to file"
            .Filter = "Constraint Files (*.con)|*.con|"
            .DefaultExt = ".con"
            .InitDir = "c:\tcs\tca\constraints"
            .Flags = cdlOFNOverwritePrompt + cdlOFNHideReadOnly
            On Error GoTo Cancel 'trap the Cancel button
20
            .ShowSave
            On Error GoTo 0 'reset
            strFN = .FileName
          End With
          Dim lngEndPos As Long
          If Len(strFN) > 0 Then
            lngEndPos = mudtFam.ActiveModel.Variables.WriteCollection(strFN, crVariableIndex,
        crVariables)
            Call mudtFam.ActiveModel.Constraints.WriteCollection(strFN, crConstraintIndex,
        lngEndPos)
30
          End If
        Cancel:
          Exit Sub
        End Sub
        Private Sub cmdPrintBatch Click()
```

	Dim blnTF As Boolean Dim udtClone As Clone
5	If mudtWord.WordApp.Documents.Count = 0 Then mudtWord.WordApp.Documents.Open FileName:=App.path & "\printing.doc" blnTF = True End If
	For Each udtClone In mudtFam.Clones mudtWord.WordApp.PrintOut FileName:=IN_DIRECTORY & udtClone.FileName Next udtClone
10	If blnTF Then mudtWord.WordApp.Documents.Close End If
	End Sub
	Private Sub cmdPrintVariants_Click()
1 <u>5</u> 1	Dim blnTF As Boolean Dim udtClone As Clone
15 mm	If mudtWord.WordApp.Documents.Count = 0 Then mudtWord.WordApp.Documents.Open FileName:=App.path & "\printing.doc" blnTF = True End If
20 11 11 11 11	For Each udtClone In mudtFam.ActiveModel.Clones mudtWord.WordApp.PrintOut FileName:=IN_DIRECTORY & udtClone.FileName Next
5 3	If blnTF Then mudtWord.WordApp.Documents.Close End If
	End Sub
	Private Sub cmdGenerate_Click()
	Dim udtClone As New Clone
30	Me.Enabled = False ' disable frmTCA to make next form seem modal frmProlog.Caption = "Generating " & txtNum2Generate & " variants" frmProlog.lblProlog.Caption = "Click Abort to terminate variant generation."

```
frmProlog.Show 'show form modeless so execution continues
          Me.MousePointer = vbHourglass
          Call mudtFam.ActiveModel.GenerateClones(mudtWord, mudtProlog,
            CInt(txtNum2Generate), sldDifference)
          Me.MousePointer = vbDefault
 5
          frmProlog.Kill 'destroy frmProlog
          Me.Enabled = True
          If lstDisposition.ListCount > 0 Then
            With lstDisposition
               .Selected(.ListCount - 1) = True
10
               Set udtClone = mudtFam.ActiveModel.Clones.Item(Str(.ItemData(.ListCount - 1)))
               Call udtClone.OpenDoc(mudtWord, IN DIRECTORY)
            End With
          End If
          UpdateTab2ControlStates
15
        End Sub
 IJ.
        Private Sub mnuDispAccept_Click()
 Ø١
 Ų٦
          Dim udtClone As Clone
          Dim nodN As Node
20:
          Dim intl As Integer
          Dim strFN As String
          With lstDisposition
            If .SelCount > 0 Then ' make sure something's selected
               For intI = 0 To .ListCount - 1 ' for multiselect
                 If .Selected(intI) Then
                   strFN =
        ExtractFileName(mudtFam.ActiveModel.Clones.Item(Str(lstDisposition.ItemData(intI))).FileNa
        me)
                   ' confirm this operation
                   If MsgBox("Accept variant " & strFN & "?",
30
                      vbQuestion + vbYesNo, "Confirm") = vbNo Then
                      .Selected(intI) = False
                   End If
                 End If
35
                 If .Selected(intI) Then
                   ' get object from active model's clone collection
                   Set udtClone = mudtFam.ActiveModel.Clones.Item(Str(.ItemData(intI)))
                   ' close the document, if it's open
                   udtClone.CloseDoc
```

	remove it from the active moder's collection
	Call mudtFam.ActiveModel.Clones.Remove(Str(.ItemData(intI)))
	' save the checksum in the model
	Call mudtFam.ActiveModel.AddChecksum(udtClone.Checksum)
5	' add it to the family clone collection
_	Call mudtFam.Clones.AddObj(udtClone)
	' add it to the accepted list box
	Call lstAccepted.AddItem(ExtractFileName(udtClone.FileName))
	' add key to itemdata
10	lstAccepted.ItemData(lstAccepted.ListCount - 1) = udtClone.index
10	' freeze the model
	mudtFam.ActiveModel.FreezeModel
	' update the icon
	Set nodN = treModels.Nodes.Item(ModelKey(mudtFam.ActiveModel.FileName))
15	nodN.Image = imSnowflake
13	stbS.Panels(pnActiveModelIcon).Picture = imII.ListImages(nodN.Image).Picture
	Call mudtFam.ActiveModel.CloseDoc
	Call mudtFam.ActiveModel.OpenDoc(mudtWord)
	End If
2Ē	Next intI
201	For intI = .ListCount - 1 To 0 Step -1
UI III	If .Selected(intI) Then
20	' remove the entry from the disposition list box
7 <u>5.</u> 8	Call .RemoveItem(intI)
25	End If
25°	Next intI
ā	End If
C)	End If End With
111	End With
- L	UpdateTab0ControlStates
3 A.	UpdateTab1ControlStates
25.	UpdateTab2ControlStates
	Opuate 1 ab2 Control States
	End Sub
	Liid Gub
	Private Sub mnuDispDefer Click()
	·
	Dim udtClone As Clone
35	Dim intl As Integer
	Dim strFN As String
	With lstDisposition
	If .SelCount > 0 Then ' make sure somethings selected
	For intI = 0 To .ListCount - 1 ' for multiselect
40	If .Selected(intI) Then

```
strFN =
        ExtractFileName(mudtFam.ActiveModel.Clones.Item(Str(lstDisposition.ItemData(intI))).FileNa
        me)
                    ' confirm this operation
                    If MsgBox("Defer variant " & strFN & "?",
 5
                      vbOuestion + vbYesNo, "Confirm") = vbNo Then
                      .Selected(intI) = False
                    End If
                 End If
10
                 If .Selected(intI) Then
                    ' get object from active model's clone collection
                    Set udtClone = mudtFam.ActiveModel.Clones.Item(Str(.ItemData(intI)))
                    ' close the document
                    udtClone.CloseDoc
15
                    ' delete the clone file
                    Kill IN DIRECTORY & udtClone.FileName
                    ' remove the clone from the active model's collection
                    Call mudtFam.ActiveModel.Clones.Remove(Str(.ItemData(intI)))
                 End If
               Next intI
               For intI = .ListCount - 1 To 0 Step -1' for multiselect
                 If .Selected(intI) Then
                    ' remove the entry from the disposition list box
                    Call .RemoveItem(intI)
                 End If
 ų)
               Next intI
            End If
          End With
          UpdateTab2ControlStates
        End Sub
        Private Sub mnuDispDiscard Click()
          Dim udtClone As Clone
          Dim intl As Integer
          Dim strFN As String
35
          With lstDisposition
            If .SelCount > 0 Then ' make sure somethings selected
               For intI = 0 To .ListCount - 1' for multiselect
                 If .Selected(intI) Then
                   strFN =
        ExtractFileName(mudtFam.ActiveModel.Clones.Item(Str(lstDisposition.ItemData(intI))).FileNa
40
```

	me)
	confirm this operation
	If MsgBox("Discard variant " & strFN & "?", _
	vbQuestion + vbYesNo, "Confirm") = vbNo Then
5	.Selected(intI) = False
	End If
	End If
	If .Selected(intI) Then
	get object from active model's clone collection
10	Set udtClone = mudtFam.ActiveModel.Clones.Item(Str(.ItemData(intI)))
	' save the checksum in the model
	Call mudtFam.ActiveModel.AddChecksum(udtClone.Checksum)
	close the document
	udtClone.CloseDoc
15	' delete the clone file
10	Kill IN DIRECTORY & udtClone.FileName
	' remove the clone from the active model's collection
	Call mudtFam.ActiveModel.Clones.Remove(Str(.ItemData(intI)))
en a	End If
20	Next intI
16 C	For intI = .ListCount - 1 To 0 Step -1' for multiselect
411	If .Selected(intI) Then
11	' remove the entry from the disposition list box
	Call .RemoveItem(intI)
2	End If
41	Next intI
	End If
C]	End With
47	
Ę.	UpdateTab2ControlStates
je f	
a L	End Sub
₽¢.	
	Private Sub mnuDispMakeModel_Click()
	•
	Dim udtClone As Clone
	Dim strNewFN As String
	Dim strKey As String
35	Dim strNewKey As String
	Dim udtM As Model
	Dim nodN As Node
	Dim intl As Integer
	Dim strFN As String
40	With 1stDisposition

```
If .SelCount > 0 Then ' make sure somethings selected
              For intI = 0 To .ListCount - 1 ' for multiselect
                If .Selected(intI) Then
                   strFN =
       ExtractFileName(mudtFam.ActiveModel.Clones.Item(Str(lstDisposition.ItemData(intI))).FileNa
 5
       me)
                   ' confirm this operation
                   If MsgBox("Create a new model from variant " & strFN & "?",
                     vbOuestion + vbYesNo, "Confirm") = vbNo Then
                     .Selected(intI) = False
10
                   End If
                 End If
                If .Selected(intI) Then
                   ' get object from active model's clone collection
15
                   Set udtClone = mudtFam.ActiveModel.Clones.Item(Str(.ItemData(intI)))
                   ' close the document
                   udtClone.CloseDoc
                   strKey = ModelKey(udtClone.FileName)
                   ' find the next key for this parent model
                   strNewKey = NextModelKey(udtClone.FileName)
                   ' add the child to the tree
                   strNewFN = ModelEmbedKey(udtClone.FileName, strNewKey)
                   Set nodN = treModels.Nodes.Add(strKey, tvwChild, strNewKey, strNewFN)
                   nodN.Expanded = True
                   nodN.sorted = True
                   nodN.Image = imSun
                   'copy the clone to the new model file name
                   Call FileCopy(IN DIRECTORY & udtClone.FileName, IN DIRECTORY &
       strNewFN)
                   ' make a copy of the parent's model file for this child
                   Call FileCopy(ModelFileName(IN DIRECTORY &
       ModelEmbedKey(udtClone.FileName, strKey)),
                     ModelFileName(IN DIRECTORY & strNewFN))
                   ' add the child's model to the model collection. "Thaw" the child.
                   Set udtM = mudtFam.Models.AddExisting(IN_DIRECTORY & strNewFN,
35
                     mudtFam.ItemType)
                   udtM.IsFrozen = False
                   ' reset the clone index of the child
                   udtM.LastClone = 0
40
                   'save it
                   udtM.WriteModel
                   ' tell 'em about it
                   Call MsgBox("Variant " & udtClone.FileName & " has been copied to " &
       strNewFN, _
                     vbInformation, "Model Created")
45
```

End If Next intI End If End With
UpdateTab0ControlStates UpdateTab2ControlStates
End Sub
Private Sub mnuFileNew_Click()
Dim udtWAPI As New Win32API Dim strFN As String
Dim udtProgram As Program Dim udtItemType As ItemType
Dim udtProximity As Proximity
Dim blnGeneric As Boolean
Dim udtIni As New IniFile
' clear out everything
ClearControls
' get family values (pun intended)
frmNew.Show vbModal
If frmNew.OK = False Then GoTo Cancel
udtProgram = frmNew.Program
udtItemType = frmNew.ItemType
udtProximity = frmNew.Proximity
blnGeneric = frmNew.Generic
With cdlCD
.InitDir = IN_DIRECTORY
.FileName = ""
.DialogTitle = "Save new family as"
.Filter = "Model Doc Files (*\$R.doc) *\$R.doc
.DefaultExt = ".doc"
.Flags = cdlOFNHideReadOnly
On Error GoTo Cancel ShowSave
On Error GoTo 0
strFN = .FileName
End With

```
' see if an FN was entered
          If Len(strFN) = 0 Then
            Beep
            GoTo Cancel
 5
          End If
          strFN = UCase(strFN)
          ' don't allow family to be created if it's not in the "IN" directory
          If InStr(1, strFN, IN DIRECTORY, vbTextCompare) Then
            ' do nothing
10
          Else
            Call MsgBox("Family must be located in " & IN DIRECTORY, _
               vbExclamation, "Error")
            GoTo Cancel
          End If
          'check the extension
15
          If (InStr(1, strFN, ".doc", vbTextCompare)) = 0 Then
            Call MsgBox("Invalid file name extension.", vbExclamation, "Error")
 ijĵ
            GoTo Cancel
          End If
201
          Dim varI As Variant
          'embed $R into FN if the user hasn't
          If InStr(1, strFN, "$R.doc", vbTextCompare) = 0 Then
            varI = InStr(1, strFN, ".doc", vbTextCompare)
            strFN = Mid(strFN, 1, varI - 1) & "$R.doc"
          End If
          'check for unique FN
          If udtWAPI.FileExists(strFN) Then
            Call MsgBox("File name " &
               ExtractFileName(strFN) & " is not unique.", _
               vbExclamation, "Error")
30
            GoTo Cancel
          End If
          Dim strShortFN As String
          strShortFN = ExtractFileName(strFN)
          ' create a new family object
35
          Set mudtFam = New Family
```

5	'set file name, program, and item type mudtFam.FileName = strFN mudtFam.Program = udtProgram mudtFam.ItemType = udtItemType mudtFam.Proximity = udtProximity mudtFam.Generic = blnGeneric mudtFam.IsDirty = True
	' put the family name on the status bar stbS.Panels(pnFamilyName) = strShortFN
10	' fill in the rest of the status bar UpdateFamilyAttributes
	' format tab 2 Call FormatTab2(mudtFam.ItemType)
15	' copy correct Word template to new model FN Select Case mudtFam.ItemType
ű U	Case ptStandardMC FileCopy App.path & "\TCASMC.doc", strFN
	Case ptQuantComp FileCopy App.path & "\TCAQC.doc", strFN
2 6	Case ptDataSuff FileCopy App.path & "\TCADS.doc", strFN
	End Select
	Dim nodN As Node
25	' clear out the treeview box treModels.Nodes.Clear
30	' add the new root Set nodN = treModels.Nodes.Add(, , "R", strShortFN, imSun nodN.Expanded = True nodN.sorted = True nodN.Selected = True
	Call mudtFam.Models.AddNew(strFN, mudtFam.ItemType)
	' enable attributes button

cmdSetAttributes.Enabled = True ' force event to set active model treModels Click Cancel: UpdateTab0ControlStates 5 Exit Sub End Sub Private Sub mnuFileOpen Click() Dim strFN As String 10 'clear out everything ClearControls With cdlCD .InitDir = IN DIRECTORY .FileName = "" .CancelError = True .DialogTitle = "Open model root" .Filter = "Model Doc Files (*\$R.doc)|*\$R.doc|" .DefaultExt = ".doc" .Flags = cdlOFNFileMustExist + cdlOFNHideReadOnly On Error GoTo Cancel .ShowOpen On Error GoTo 0 strFN = .FileNameEnd With 'exit if there's no file name If Len(strFN) = 0 Then Exit Sub End If 30 strFN = UCase(strFN)' don't allow family to be opened if it's not in the "IN" directory If InStr(1, strFN, IN DIRECTORY, vbTextCompare) Then ' do nothing Else

```
Call MsgBox("Family must be located in " & IN DIRECTORY,
               vbExclamation, "Error")
            Exit Sub
          End If
 5
          ' find all of the children
          Dim nodN As Node
          Dim strIndex As String
          Dim strT As String
          Dim varI1 As Variant
          Dim udtWAPI As New Win32API
10
          Dim strNewFN As String
          Dim colFN As Collection
          ' add a wild card to the file name
          varI1 = InStr(1, strFN, ".")
          strNewFN = Mid(strFN, 1, varI1 - 1) & "*" & Mid(strFN, varI1, _
15
            Len(strFN) - varI1 + 1)
          'get a collection of file names (*.doc) matching the wild card
 ij.
          Set colFN = udtWAPI.FindAllFiles(strNewFN)
 đ١
 Ų1
          ' create a new family object
          Set mudtFam = New Family
205
 IJ,
          Dim strMdfFN As String
          ' make sure the .mdf file is there.
          strMdfFN = left(strFN, Len(strFN) - 3) & "mdf"
          If udtWAPI.FileExists(strMdfFN) = False Then
            Call MsgBox("This family has a " &
               "missing mdf file and cannot be loaded. " &
               "File " & strMdfFN & " is not in the IN directory.", _
               vbExclamation, "Error")
            Exit Sub
30
          End If
          ' set the file name of the family, read.
          mudtFam.FileName = strFN
          mudtFam.ReadFamily
          Dim udtClone As Clone
          'verify that all variants referenced in the family object are in
35
          ' the IN directory.
```

	'the next line allows families to be renamed between TCA sessions udtClone.FileName = ExtractFamilyName(strFN) &
5	ExtractFamilyKey(udtClone.FileName) & ".doc" If udtWAPI.FileExists(IN_DIRECTORY & udtClone.FileName) = False Then Call MsgBox("This family has at least " & _ "one missing variant file and cannot be loaded. " & _ "File " & udtClone.FileName & " is not in the IN directory.", _ vbExclamation, "Error")
10	Exit Sub End If Next udtClone
	' put family name on status bar stbS.Panels(pnFamilyName) = ExtractFileName(strFN)
15	' format tab 2 Call FormatTab2(mudtFam.ItemType)
20 1.35 1.35 1.35 1.35 1.35 1.35 1.35 1.35	' update the accepted listbox with leftover clones For Each udtClone In mudtFam.Clones With lstAccepted If udtClone.IsRouted Then Call .AddItem(udtClone.FileName & ": Routed to TCS") Else Call .AddItem(udtClone.FileName) End If .ItemData(.ListCount - 1) = udtClone.index End With Next udtClone ' select the first entry, if there is one If lstAccepted.ListCount > 0 Then lstAccepted.Selected(0) = True End If
	' display attribute info on status bar UpdateFamilyAttributes
35	' clear out the dummy list box Call lstDummy.Clear
	Dim varFN As Variant Dim udtM As Model Dim intI As Integer

Dim intIcon As Integer

nodN.sorted = True

40

```
' the tree control must add them in heirarchical order.
          For Each varFN In colFN
            varI1 = InStr(1, varFN, ".")
 5
            If IsNumeric(Mid(varFN, varI1 - 1, 1)) = False Then 'it's not a clone
               Call lstDummy.AddItem(varFN) ' add the model
            End If
          Next varFN
10
          Dim strMdlFN As String
          For intI = 0 To lstDummy.ListCount - <math>1
            varFN = lstDummy.List(intI)
            strIndex = ModelKey(varFN)
            If UCase(strIndex) = "R" Then
               Set nodN = treModels.Nodes.Add(, , strIndex, varFN)
              Set treModels.SelectedItem = nodN
            Else
               Set nodN = treModels.Nodes.Add(left(strIndex, Len(strIndex) - 1), _
                 tvwChild, strIndex, varFN)
 ij
            End If
20.
 ų)
            ' test to see if corresponding .mdl file exists
            strMdlFN = left(varFN, Len(varFN) - 3) & "mdl"
            If udtWAPI.FileExists(strMdlFN) = False Then
               Call MsgBox("This family has at least " &
                 "one missing mdl file and cannot be loaded. " &
                 "File " & strMdlFN & " is not in the IN directory.",
                 vbExclamation, "Error")
              ClearControls
              Exit Sub
30
            End If
            ' add a new model to the collection
            Set udtM = mudtFam.Models.AddExisting(IN DIRECTORY & varFN,
               mudtFam.ItemType)
            If udtM.IsFrozen Then
               nodN.Image = imSnowflake
35
            Else
              nodN.Image = imSun
            End If
            nodN.Expanded = True
```

'dump the file names into a dummy list box which will sort them automatically.

```
Next intI
          ' enable attributes button
          cmdSetAttributes.Enabled = True
          ' force event to set active model
 5
          treModels Click
        Cancel:
          Update Tab 0 Control States \\
          Exit Sub
        End Sub
        Private Sub mnuFileImportItem_Click()
10
          Dim udtIni As New IniFile
          Dim strFN As String
 T.
          ' clear out everything
          ClearControls
          With cdlCD
             .InitDir = IN DIRECTORY
             .FileName = \overline{}""
             .CancelError = True
             .DialogTitle = "Open locked item"
             .Filter = "Item Doc Files (*.doc)|*.doc|"
             .DefaultExt = ".doc"
 C)
             .Flags = cdlOFNFileMustExist + cdlOFNHideReadOnly
             On Error GoTo Cancel
25
             .ShowOpen
             On Error GoTo 0
             strFN = .FileName
          End With
        ' End If
30
          ' exit if there's no file name
          If Len(strFN) = 0 Then
             Exit Sub
          End If
          ' don't allow locked item to be opened if it's not in the "IN" directory
```

VBSCA -179-

```
If InStr(1, strFN, IN DIRECTORY, vbTextCompare) Then
            ' do nothing
          Else
            Call MsgBox("Locked item must be located in " & IN DIRECTORY, _
              vbExclamation, "Error")
 5
            Exit Sub
         End If
          ' set the FN of the ini that accompanies the locked item
          udtIni.FN = IN DIRECTORY & ExtractFileNameNoExt(strFN) & ".ini"
10
          Dim udtW As New Win32API
         If udtW.FileExists(udtIni.FN) = False Then
            Call MsgBox("Ini file must accompany locked item " & ExtractFileName(strFN) & _
              ".", vbExclamation, "Error")
            Exit Sub
15
         End If
         Dim udtProgram As Program
 ű
          Dim udtDeliveryMode As DeliveryMode
 ۵ì
          Dim udtItemType As ItemType
 Li
          Dim strAccNum As String
 Ŵ
          ' find out about this locked item from the .ini file
20:
 Ų)
          Select Case udtIni.GetProfileString("LockedItemData", "Program")
            Case "GRE"
              udtProgram = prGRE
            Case "GMAT"
              udtProgram = prGMAT
            Case "SAT"
              udtProgram = prSAT
            Case "Not Found"
              Call MsgBox("No Program entry found in ini file " & ExtractFileName(strFN) &
30
                 ".", vbExclamation, "Error")
              Exit Sub
          End Select
          Select Case udtIni.GetProfileString("LockedItemData", "DeliveryMode")
            Case "CBT"
35
              udtDeliveryMode = dmCBT
            Case "PPT"
              udtDeliveryMode = dmPPT
            Case "Not Found"
              Call MsgBox("No DeliveryMode entry found in ini file " & ExtractFileName(strFN) & _
```

```
".", vbExclamation, "Error")
              Exit Sub
          End Select
          Select Case udtIni.GetProfileString("LockedItemData", "ItemType")
            Case "MC Item", "QantDisc", "MC", "Multiple Choice"
 5
              udtItemType = ptStandardMC
            Case "DataSuff", "DS", "Data Sufficiency"
              udtItemType = ptDataSuff
            Case "QC Discrete", "QantComp", "QC", "Quantitative Comparison"
              udtItemType = ptQuantComp
10
            Case "Not Found"
              Call MsgBox("No ItemType entry found in ini file " & ExtractFileName(strFN) &
                 ".", vbExclamation, "Error")
              Exit Sub
15
          End Select
          strAccNum = udtIni.GetProfileString("LockedItemData", "LockedAccnum")
          If strAccNum = "Not Found" Then strAccNum = ""
 Ú
          'initialize locked item object
 đί
          Dim udtLI As New LockedItem
 Ľ1
201
          udtLI.LockedItemFileName = strFN
          udtLI.WordInstance = mudtWord
         If udtLI.OpenLockedItemDoc = False Then 'we couldn't figure out what doc and item type it
       was
            Call MsgBox("Locked item file appears to be damaged.", vbExclamation, "Error")
            udtLI.CloseLockedItemDoc
            Exit Sub
          End If
          With cdlCD
            .FileName = ""
            .DialogTitle = "Save new family based on this locked item as"
30
            .Filter = "Model Doc Files (*$R.doc)|*$R.doc|"
            .DefaultExt = ".doc"
            .Flags = cdlOFNHideReadOnly
            On Error GoTo CloseAndCancel
            .ShowSave
35
            On Error GoTo 0
            strFN = .FileName
          End With
         End If
```

```
' see if an FN was entered
          If Len(strFN) = 0 Then
            Beep
            Exit Sub
 5
          End If
          strFN = UCase(strFN)
          'check the extension
          If (InStr(1, strFN, ".doc", vbTextCompare)) = 0 Then
            Call MsgBox("Invalid file name extension.", vbExclamation, "Error")
10
            Exit Sub
          End If
          Dim varI As Variant
          'embed $R into FN if the user hasn't
          If InStr(1, strFN, "$R.doc", vbTextCompare) = 0 Then
            varI = InStr(1, strFN, ".doc", vbTextCompare)
            strFN = Mid(strFN, 1, varI - 1) \& "$R.doc"
          End If
 'check for unique FN
          Dim udtWAPI As New Win32API
20
          If udtWAPI.FileExists(strFN) Then
            Call MsgBox("File name " &
               ExtractFileName(strFN) & " is not unique.", _
               vbExclamation, "Error")
            Exit Sub
          End If
          'copy the ini file of the locked item to the family name
          Call FileCopy(udtIni.FN, left(strFN, Len(strFN) - 3) & "ini")
          Dim strShortFN As String
          strShortFN = ExtractFileName(strFN)
30
          'create a new family object
          Set mudtFam = New Family
          'put family name on status bar
          stbS.Panels(pnFamilyName) = strShortFN
          'set file name, program, and item type
```

5	mudtFam.FileName = strFN mudtFam.Program = udtProgram mudtFam.ItemType = udtItemType mudtFam.AccNum = strAccNum mudtFam.IsDirty = True
	' format tab 2 Call FormatTab2(mudtFam.ItemType)
	' copy correct Word template to new model FN Select Case mudtFam.ItemType
10	Case ptStandardMC FileCopy App.path & "\TCASMC.doc", strFN
	Case ptQuantComp FileCopy App.path & "\TCAQC.doc", strFN
15	Case ptDataSuff FileCopy App.path & "\TCADS.doc", strFN
	End Select
1.5 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0	Dim nodN As Node
	' clear out the treeview box treModels.Nodes.Clear
2 H. H. H. H. H.	' add the new root Set nodN = treModels.Nodes.Add(, , "R", strShortFN, imSun) nodN.Expanded = True nodN.sorted = True nodN.Selected = True
25	Call mudtFam.Models.AddNew(strFN, mudtFam.ItemType)
	mudtFam.Generic = False mudtFam.Proximity = prNear
	'enable attributes button cmdSetAttributes.Enabled = True
30	' force event to set attributes cmdSetAttributes_Click

	' force event to set active model
	treModels_Click
	C.1. C. N. T
	Select Case udtItemType
-	Case ptStandardMC
5	Select Case udtDeliveryMode
	Case dmCBT
	Call udtLI.ConvertCBTSMCItem
	Case dmPPT
10	Call udtLI.ConvertPPTSMCItem
10	End Select
	Case ptDataSuff
	Call udtLI.ConvertDSItem
	Case ptQuantComp
15	Select Case udtDeliveryMode Case dmCBT
13	Case diffCB1 Call udtLI.ConvertCBTQCItem
	Case dmPPT
,	Call udtLI.ConvertPPTQCItem
	End Select
26	End Select
2 <u>9</u> 1	End Select
₩ i FF	CloseAndCancel:
.in	Crosser indoduces.
	udtLI.CloseLockedItemDoc
8	Cancel:
Ē	
4J	UpdateTab0ControlStates
	•
25,	Exit Sub
THE STATE OF THE S	End Sub
	District C. Language File Forth Clinks
	Private Sub mnuFileExit_Click()
	Call Form_Unload(0)
	End
30	End Sub
	'Private Sub ReturnToTab0()
	•
	' Dim intPrevTab As Integer

	If sstMainTab.Tab = 0 Then Exit Sub
5	intPrevTab = sstMainTab.Tab sstMainTab.Tab = 0 Call sstMainTab_Click(intPrevTab)
	'End Sub
10	Private Sub mnuFilePrintSetup_Click()
10	cdlCD.Flags = cdlPDPrintSetup
15	On Error GoTo Cancel cdlCD.ShowPrinter On Error GoTo 0
	Cancel:
	Exit Sub
	End Sub
13 mg C3 mm r. a mg a k mg 20 mg mg mg mg mg	Private Sub mnuHelpAbout_Click()
	frmAbout.Show vbModal
	End Sub
	Private Sub mnuTreeExtend_Click()
n (3''' 2"'' 1"'' 1" 1" 1" 25''	Dim nodN As Node Dim strFN As String Dim strNewFN As String Dim strKey As String Dim strT As String Dim strNewKey As String
30	If treModels.SelectedItem Is Nothing Then Exit Sub
	Set nodN = treModels.SelectedItem strFN = nodN.Text
35	' confirm this operation If MsgBox("Make a child model from model " & strFN & "?", _ vbQuestion + vbYesNo, "Confirm") = vbNo Then

```
Exit Sub
         End If
         strKey = ModelKey(strFN)
         strNewKey = NextModelKey(strFN)
          ' add the child to the tree
 5
         strNewFN = ModelEmbedKey(strFN, strNewKey)
         Set nodN = treModels.Nodes.Add(strKey, tvwChild, strNewKey, strNewFN)
         nodN.Expanded = True
         nodN.sorted = True
10
         nodN.Image = imSun
          'deactivate active model to close it before file copies, if the active
          ' model is being extended.
         Dim blnReopenModel As Boolean
         blnReopenModel = False
         If strFN = stbS.Panels(pnActiveModelName) Then
155
            Call mudtFam.ActiveModel.CloseDoc
 Minghest Start Ben Con-
            blnReopenModel = True
         End If
         ' make a copy of the parent's word doc for this child
         Call FileCopy(IN DIRECTORY & strFN, IN DIRECTORY & strNewFN)
          ' make a copy of the parent's model file for this child
         Call FileCopy(IN_DIRECTORY & ModelFileName(strFN), IN_DIRECTORY &
       ModelFileName(strNewFN))
          ' add the child's model to the model collection. "Thaw" the child.
         Dim udtM As Model
25
         Set udtM = mudtFam.Models.AddExisting(IN DIRECTORY & strNewFN, _
            mudtFam.ItemType)
         udtM.IsFrozen = False
          ' reset the clone index of the child
         udtM.LastClone = 0
30
          ' reset the checksums
         udtM.InitChecksums
          'save it
```

	If blnReopenModel Then Call mudtFam.ActiveModel.OpenDoc(mudtWord) End If
5	End Sub
	Private Sub mnuTreeRemove_Click()
	Dim nodN As Node Dim strFN As String Dim strKey As String
10	If treModels.SelectedItem Is Nothing Then Exit Sub
	Set nodN = treModels.SelectedItem strFN = nodN.Text
	strKey = ModelKey(strFN)
5 7	Dim colIndices As New Collection
15.	' don't remove if this node or any descendant nodes are frozen Dim udtModel As Model
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	'check selected node If treModels.SelectedItem.index = 1 Then 'it's the root model Call MsgBox("The root model can't be removed.", vbExclamation, "Error" Exit Sub End If
	Set udtModel = mudtFam.Models.Item(treModels.SelectedItem) If udtModel.IsFrozen Then Call MsgBox("Can't remove frozen model.", vbExclamation, "Error")
25	Exit Sub Else Call colIndices.Add(treModels.SelectedItem.index) End If
30	Dim blnDone As Boolean blnDone = False
	' check if any of it's descendants are frozen

udtM.WriteModel

Do

```
Set nodN = nodN.Child
            If nodN Is Nothing Then
              ' do nothing
            Else
 5
              Do
                If mudtFam.Models.Item(nodN.Text).IsFrozen Then
                   Call MsgBox("Can't remove model with one or more frozen descendants.",
                     vbExclamation, "Error")
                   Exit Sub
10
                 End If
                 Call colIndices.Add(nodN.index)
              Loop Until nodN.index = nodN.LastSibling.index
            End If
          Loop Until nodN Is Nothing
15
          ' confirm this operation
          If MsgBox("Remove model " & strFN & " and it's children?",
            vbQuestion + vbYesNo, "Confirm") = vbNo Then
            Exit Sub
          End If
          'close active model document as we're deleting it
          mudtFam.ActiveModel.CloseDoc
          mudtFam.ActiveModel = Nothing
          stbS.Panels(pnActiveModelIcon).Picture = Nothing
          stbS.Panels(pnActiveModelName) = ""
25
          Dim varIndex As Variant
          'remove all effected models from the family
          For Each varIndex In colIndices
            Call mudtFam.Models.Remove(treModels.Nodes(varIndex))
            Kill IN DIRECTORY & left(treModels.Nodes(varIndex),
              Len(treModels.Nodes(varIndex)) - 3) & "*"
30
          Next varIndex
          ' remove them from the tree control
          Call treModels.Nodes.Remove(colIndices(1))
       End Sub
       Private Sub mnuVariablesAdd Click()
35
          frmVariable.AddEditFlag = aeAdd
```

```
End Sub
       Private Sub mnuVariablesEdit Click()
          frmVariable.AddEditFlag = aeEdit
       End Sub
       Private Sub mnuVariablesRemove_Click()
          Dim intInd As Integer
          intInd = lstVariables.ListIndex 'Get index
          'Make sure list item is selected
          If intInd < 0 Then
10
            Beep
            Exit Sub
          End If
          Dim strVN As String
          strVN = mudtFam.ActiveModel.Variables.Item(Str(lstVariables.ItemData(intInd))).name
15.
          'confirm this operation
          If MsgBox("Remove variable " & strVN & "?",
            vbQuestion + vbYesNo, "Confirm") = vbNo Then
            Exit Sub
          End If
          'Remove the variable from the collection using the key in the list box
          Call mudtFam.ActiveModel.Variables.Remove(Str(lstVariables.ItemData(intInd)))
          'Remove the variable from the list box
          Call lstVariables.RemoveItem(intInd)
          UpdateTab1ControlStates
25
       End Sub
        'Empty the variable list box
       Private Sub mnuVariablesRemoveAll Click()
          'confirm this operation
          If MsgBox("Remove all variables?", _
```

	vbQuestion + vbYesNo, "Confirm") = vbNo Then Exit Sub End If
5	'clear the list box IstVariables.Clear
	' empty the collection mudtFam.ActiveModel.Variables.Clear
	UpdateTab1ControlStates
	End Sub
10	Private Sub mnuVariablesEnableAll_Click()
	Call SetAllCheckboxes(True)
C)	UpdateTab1ControlStates
	End Sub
### ##################################	Private Sub mnuVariablesDisableAll_Click()
1 5 2	Call SetAllCheckboxes(False)
≒. E F1	UpdateTab1ControlStates
	End Sub
	Private Sub mnuVariablesTest_Click()
	Call TestConstraints(tcTestVariables)
20	End Sub Private Sub mnuConstraintsAdd_Click()
25	'set the add flag for frmConstraints frmConstraints.AddEditFlag = aeAdd 'set the list box frmConstraints.ListBox = lstConstraints(mintConstrLBInd) 'set the model frmConstraints.Model = mudtFam.ActiveModel 'set the constraint type frmConstraints.ConstraintType = mintConstrLBInd

```
Call UpdateTab1ControlStates(mintConstrLBInd)
       End Sub
       Private Sub mnuConstraintsEdit_Click()
 5
          If lstConstraints(mintConstrLBInd).ListIndex >= 0 Then 'Make sure list item is selected
            ' set the edit flag for frmConstraints
            frmConstraints.AddEditFlag = aeEdit
            ' set the list box
            frmConstraints.ListBox = lstConstraints(mintConstrLBInd)
10
            ' set the model
            frmConstraints.Model = mudtFam.ActiveModel
            ' set the constraint
            With lstConstraints(mintConstrLBInd)
               frmConstraints.Constraint =
                 mudtFam.ActiveModel.Constraints.Item(Str(.ItemData(.ListIndex)))
            End With
            ' set the constraint type
            frmConstraints.ConstraintType = mintConstrLBInd
            ' crank up the form
201
            frmConstraints.Show vbModal
          Else
            Beep
          End If
          Call UpdateTab1ControlStates(mintConstrLBInd)
       End Sub
       Private Sub mnuConstraintsRemove Click()
          Dim intInd As Integer
          intInd = lstConstraints(mintConstrLBInd).ListIndex 'Get index
30
          ' Make sure list item is selected
          If intInd < 0 Then
            Beep
            Exit Sub
          End If
```

' crank up the form

frmConstraints.Show vbModal

```
Dim udtCon As Constraint
          Set udtCon =
        mudtFam.ActiveModel.Constraints.Item(Str(lstConstraints(mintConstrLBInd).ItemData(intInd))
 5
        )
          ' confirm this operation
          If MsgBox("Remove constraint " & udtCon.ConstraintString & "?",
            vbQuestion + vbYesNo, "Confirm") = vbNo Then
            Exit Sub
          End If
10
          'Remove the variable from the collection using the key in the list box
          Call
        mudtFam.ActiveModel.Constraints.Remove(Str(lstConstraints(mintConstrLBInd).ItemData(intI
        nd)))
15
          'Remove the variable from the list box
          Call lstConstraints(mintConstrLBInd).RemoveItem(intInd)
 櫚
          Call UpdateTab1ControlStates(mintConstrLBInd)
 Mr. Man Mr. Sea Com
        End Sub
        Private Sub mnuConstraintsRemoveAll Click()
            ' confirm this operation
          If MsgBox("Remove all constraints in this list box?",
            vbQuestion + vbYesNo, "Confirm") = vbNo Then
             Exit Sub
          End If
          'clear the list box
          lstConstraints(mintConstrLBInd).Clear
          'empty the collection
          Call mudtFam.ActiveModel.Constraints.Clear(mintConstrLBInd)
          Call UpdateTab1ControlStates(mintConstrLBInd)
30
        End Sub
        Private Sub mnuConstraintsEnableAll Click()
          Call SetAllCheckboxes(True)
```

End Sub Private Sub mnuConstraintsDisableAll Click() Call SetAllCheckboxes(False) 5 Call UpdateTab1ControlStates(mintConstrLBInd) End Sub Private Sub mnuConstraintsTest Click() cmdSaveModel Click ' force a save Select Case mintConstrLBInd 10 Case ctVariation Call TestConstraints(tcTestVariationConstraints) Case ctDistractor Call TestConstraints(tcTestDistractorConstraints) **End Select** 15] End Sub Private Sub mnuAcceptedProfile_Click() Dim udtClone As Clone Dim intI As Integer ' set the family frmDifficulty.Family = mudtFam ' set the clone With lstAccepted For intI = 0 To .ListCount - 1 If .Selected(intI) Then Set udtClone = 25 mudtFam.Clones.Item(Str(.ItemData(intI))) frmDifficulty.Clone = udtClone Exit For End If Next intI 30 End With

Call UpdateTab1ControlStates(mintConstrLBInd)

```
'give frmDifficulty a caption
                               frmDifficulty.Caption = "Profile of variant " & _
                                     ExtractFileName(udtClone.FileName)
                              ' crank up the form
   5
                              frmDifficulty.Show vbModal
                              If udtClone.IsRouted Then
                                     lstAccepted.List(intI) = udtClone.FileName & ": Routed to TCS"
                              Else
                                     lstAccepted.List(intI) = udtClone.FileName
10
                              End If
                       End Sub
                       Private Sub mnuAcceptedCopy_Click()
                              Dim udtClone As Clone
                               'this menu option is only active if a variant with a completed profile
                              ' is currently selected.
15
   Hing office of the state of the
                               With lstAccepted
                                     Set udtClone = mudtFam.Clones.Item(Str(.ItemData(.ListIndex)))
                              End With
                              'copy necessary stuff into a holding area
                              Set mudtClone = udtClone
                              UpdateTab0ControlStates
                       End Sub
                       'this menu option is only active if a profile has been copied
                       Private Sub mnuAcceptedPaste Click()
25
                              Dim udtClone As Clone
                              Dim intI As Integer
                               With lstAccepted
                                     If .SelCount > 0 Then
                                             ' confirm this operation
                                            If MsgBox("Paste profile of variant " & mudtClone.FileName & _
30
                                                    " to all selected variants?", _
                                                   vbQuestion + vbYesNo, "Confirm") = vbNo Then
                                                    Exit Sub
```

```
End If
               For intI = 0 To .ListCount - 1
                 If .Selected(intI) Then
                    Set udtClone = mudtFam.Clones.Item(Str(.ItemData(intI)))
                    'copy necessary stuff from the holding area
 5
                    udtClone.Domain = mudtClone.Domain
                    udtClone.BatchID = mudtClone.BatchID
                    udtClone.DeliveryMode = mudtClone.DeliveryMode
                    udtClone.Nature = mudtClone.Nature
                    udtClone.IsRouted = mudtClone.IsRouted
10
                    udtClone.TDEstimate = mudtClone.TDEstimate
                    udtClone.IsDifficultyCalculated = mudtClone.IsDifficultyCalculated
                    If udtClone.IsDifficultyCalculated Then
                      udtClone.DiffEst = mudtClone.DiffEst.Copy
15
                    End If
                    If udtClone.IsRouted Then
                      .List(intI) = udtClone.FileName & ": Routed to TCS"
                    Else
                      .List(intI) = udtClone.FileName
رِي
20
                    End If
                 End If
 ۵ì
               Next intI
            End If
          End With
25
        End Sub
 C)
        'checks/unchecks all checkboxes in a listbox and enable/disable their
 ű
        'associated variable or constraint objects
        Private Sub SetAllCheckboxes(ByVal blnBool As Boolean)
          Dim i As Integer
          For i = 0 To (mlstCurrentListBox.ListCount - 1)
30
            mlstCurrentListBox.Selected(i) = blnBool
          Next i
          Dim udtV As Variable
          Dim udtC As Constraint
35
          If mlstCurrentListBox.name = "lstVariables" Then
            For Each udtV In mudtFam.ActiveModel.Variables
               udtV.Enabled = blnBool
            Next udtV
```

```
Else
            For i = 0 To (mlstCurrentListBox.ListCount - 1)
               Set udtC =
        mudtFam.ActiveModel.Constraints.Item(Str(mlstCurrentListBox.ItemData(i)))
 5
               udtC.Enabled = blnBool
            Next i
          End If
        End Sub
       Private Sub mwudtModelTest PrologFinished()
10
        End Sub
        Private Sub sstMainTab Click(PreviousTab As Integer)
          Static blnRecursing As Boolean
          Static bytMessage As Byte
          If blnRecursing Then
 ď]
151
            Select Case bytMessage
               Case 1
                 Call MsgBox("Open a model family using the File menu.",
                   vbExclamation, "Error")
               Case 2
                 Call MsgBox("Set the active model by clicking on a model.", __
                   vbExclamation, "Error")
            End Select
            blnRecursing = False
            Exit Sub
          End If
2₫;
          'error conditions
          If sstMainTab.Tab > 0 Then
            If treModels.Nodes.Count = 0 Then ' family hasn't been set
               bytMessage = 1
               blnRecursing = True
30
               sstMainTab.Tab = PreviousTab ' will trigger recursion
               Exit Sub
            End If
          End If
35
          If sstMainTab.Tab = 1 Or sstMainTab.Tab = 2 Then
            If mudtFam.ActiveModel Is Nothing Then 'active model has not been set
               bytMessage = 2
```

```
blnRecursing = True
               sstMainTab.Tab = PreviousTab ' will trigger recursion
               Exit Sub
            End If
 5
          End If
          ' if we got here, everything's ok!
          If PreviousTab = 2 Then
            txtNum2Generate = ""
          End If
10
          If PreviousTab = 1 Then
            If mudtFam.ActiveModel.IsDirty Then
               KillVariants 'delete any variants on tab 3
               mudtFam.ActiveModel.InitTempChecksums 'initialize temp checksums
            End If
15
          End If
          ' save family
          mudtFam.WriteFamily
 L.
 O
          ' save the active model
          If mudtFam.ActiveModel Is Nothing Then
201
            ' do nothing
          Else
            mudtFam.ActiveModel.WriteModel
          End If
          Select Case sstMainTab.Tab
            Case 0
              'enable new/open
              cmdSetAttributes.Default = True
              mnuFileNew.Enabled = True
              mnuFileOpen.Enabled = True
              mnuFileImportItem.Enabled = True
30
              If PreviousTab = 2 Then
                 mudtFam.ActiveModel.CloseAllCloneDocs
                 Call mudtFam.ActiveModel.OpenDoc(mudtWord)
              End If
35
              ' if there are no variants, disable the print button
              If lstAccepted.ListCount > 0 Then
                 cmdPrintBatch.Enabled = True
              Else
```

	Case 1
	cmdSaveModel.Default = True
5	' disable new/open
	mnuFileNew.Enabled = False
	mnuFileOpen.Enabled = False
	mnuFileImportItem.Enabled = False
	'warn if variants exist in lstDisposition and model isn't frozen
10	If mudtFam. ActiveModel. Is Frozen = False Then
	If lstDisposition.ListCount > 0 Then 'variants exist
	Call MsgBox("Variants on tab 3 will be deleted if " & _
	"the model is changed.", vbInformation, "Warning")
	End If
15	End If
	If $PreviousTab = 0$ Then
	mudtFam.CloseAllCloneDocs
£1	Call mudtFam.ActiveModel.OpenDoc(mudtWord)
41	End If
2₫1	If $PreviousTab = 2$ Then
Ľ1	mudtFam.ActiveModel.CloseAllCloneDocs
and a	Call mudtFam.ActiveModel.OpenDoc(mudtWord)
41	End If
de fi	
1 2,7	Case 2
2 5	cmdGenerate.Default = True
12.	' disable new/open
	mnuFileNew.Enabled = False
j ek	mnuFileOpen.Enabled = False
	mnuFileImportItem.Enabled = False
<u>[]</u> 30	! disable the concrete button
30	' disable the generate button ' cmdGenerate.Enabled = False
	emdenerate.Enabled – Paise
	' if there are no variants, disable the print button
	If lstDisposition.ListCount > 0 Then
	cmdPrintVariants.Enabled = True
35	Else
	cmdPrintVariants.Enabled = False
	End If
	If $PreviousTab = 0$ Then
	mudtFam.CloseAllCloneDocs
40	End If

cmdPrintBatch.Enabled = False

End If

```
' display the currently selected document
              With lstDisposition
                If .ListCount > 0 Then 'a valid selection has been made
                   Call mudtFam.ActiveModel.Clones.Item
                     (Str(.ItemData(.ListIndex))).OpenDoc(mudtWord, IN_DIRECTORY)
 5
                Else
                   Call mudtFam.ActiveModel.OpenDoc(mudtWord)
                End If
              End With
          End Select
10
       End Sub
       ' restore full window drag, if necessary
       Private Sub sstMainTab MouseMove(Button As Integer, _
          Shift As Integer, X As Single, Y As Single)
          Dim udtW As Win32API
          If mblnRestoreFullWindowDrag Then
 ជា
            Set udtW = New Win32API
            udtW.TurnOnFullWindowDrag
 ĽĴ
            mblnRestoreFullWindowDrag = False
20:
         End If
          If mudtWord Is Nothing Then Exit Sub
          If sstMainTab.Tab = 1 Then 'do this first, as there will be an active doc
            If mudtWord.WordApp.ActiveDocument.Saved = False And _
              cmdSaveModel.Enabled = False Then
              If Not mudtFam.ActiveModel.IsFrozen Then
                mudtFam.ActiveModel.IsDirty = True
                UpdateTab1ControlStates
              End If
30
            End If
          End If
       End Sub
       Private Sub treModels_Click()
          Dim nodN As Node
```

If treModels.SelectedItem Is Nothing Then Exit Sub-Set nodN = treModels.SelectedItem 'put model icon and name on status bar stbS.Panels(pnActiveModelIcon).Picture = imlI.ListImages(nodN.Image).Picture stbS.Panels(pnActiveModelName) = treModels.SelectedItem 'close doc for existing active model If mudtFam.ActiveModel Is Nothing Then ' do nothing Else mudtFam.ActiveModel.CloseDoc End If ' set the new active model and activate it mudtFam.ActiveModel = mudtFam.Models.Item(treModels.SelectedItem) Call mudtFam.ActiveModel.OpenDoc(mudtWord) ' clear out the Variable list box lstVariables.Clear 'populate the variable list box with this model's variables Dim udtVar As Variable For Each udtVar In mudtFam.ActiveModel.Variables With lstVariables Call .AddItem(udtVar.ScreenFormat) .ItemData(.ListCount - 1) = udtVar.index.Selected(.ListCount - 1) = udtVar.Enabled End With Next udtVar Dim intI ' clear out the constraint list boxes lstConstraints(0).Clear lstConstraints(1).Clear 'populate the constraint list boxes with this model's constraints Dim udtCon As Constraint For Each udtCon In mudtFam.ActiveModel.Constraints

5

10

1**5**]

Œ١

Her albo His alba

30

intI = udtCon.ConstraintType
With lstConstraints(intI)

```
.ItemData(.ListCount - 1) = udtCon.index
              .Selected(.ListCount - 1) = udtCon.Enabled
            End With
 5
          Next udtCon
          'populate comments form
          frmComments.Comment = mudtFam.ActiveModel.Comments
          ' clear out the clone disposition list box
          lstDisposition.Clear
10
          'populate the clone list box with this model's clones
          Dim udtClone As Clone
          With lstDisposition
            For Each udtClone In mudtFam.ActiveModel.Clones
              Call .AddItem(ExtractFileName(udtClone.FileName))
              .ItemData(.ListCount - 1) = udtClone.index
            Next udtClone
          End With
          ' save the active model
          mudtFam.ActiveModel.WriteModel
          'adjust menu/button states depending on active model properties
20
          UpdateTab1ControlStates
          UpdateTab2ControlStates
          'enable extend
          mnuTreeExtend.Enabled = True
       End Sub
       Private Sub treModels MouseUp(Button As Integer, Shift As Integer, _
          X As Single, Y As Single)
          If treModels.Nodes.Count > 0 Then
            If Button = vbRightButton Then
              PopupMenu mnuTree
30
            End If
          End If
       End Sub
```

Call .AddItem(udtCon.ConstraintString)

```
Private Sub txtNum2Generate_Change()
          If Val(txtNum2Generate) > 0 Then
            cmdGenerate.Enabled = True
          Else
 5
            cmdGenerate.Enabled = False
          End If
       End Sub
       Private Sub txtVariablize GotFocus()
          If mudtWord.DocumentsCount = 0 Then
10
            Beep
          Else
            If mudtWord.SelectionType < wdSelectionNormal Then
              Call MsgBox("Nothing is selected.", vbExclamation, "Error")
            Else
              Call AddUndefinedVariables(mudtWord.SelectionText)
            End If
 4]
          End If
20°
       End Sub
       'scans a string for undefined variable names and add them to
       ' the variable collection and list box
       Public Sub AddUndefinedVariables(ByVal strNames As String)
         Dim colC As Collection
          Dim strS As Variant
         Dim udtVar As Variable
          Dim colDummy As New Collection
          Set colC = UndefinedNames(strNames)
          ' don't do it if the model is frozen!
         If Not mudtFam Is Nothing Then
            If Not mudtFam.ActiveModel Is Nothing Then
30
              If mudtFam.ActiveModel.IsFrozen Then
                Call MsgBox("Variables cannot be added to a frozen model.", _
                   vbExclamation, "Error")
                 Exit Sub
35
              End If
            End If
```

End If

For Each strS In colC If MsgBox("Auto-define variable " & strS & "?", vbQuestion + vbYesNo, _ "New variable detected") = vbYes Then Select Case left(strS, 1) 5 Case "I" Set udtVar = mudtFam.ActiveModel.Variables.AddInteger(strS, _ True, "1", "100", "1", False, True) Set udtVar = mudtFam.ActiveModel.Variables.AddReal(strS, 10 True, "1", "100", "1", False, True, True, ".01", True) Set udtVar = mudtFam.ActiveModel.Variables.AddString(strS, _ True, True, Chr(164), True, colDummy) 15 Case "F" Set udtVar = mudtFam.ActiveModel.Variables.AddFraction(strS, True, "1", "1", "100", "1", "1", ",1", False, True, False) ű m U Set udtVar = mudtFam.ActiveModel.Variables.AddUntyped(strS, 20: True, False) Case Else 'assume untyped Set udtVar = mudtFam.ActiveModel.Variables.AddUntyped(strS, True, False) End Select With lstVariables ' Add the new variable to the variable list box Call .AddItem(udtVar.ScreenFormat) 'Set ItemData to index value of the variable object .ItemData(.ListCount - 1) = udtVar.index30 'Check the check box .Selected(.ListCount - 1) = True End With End If Next strS 35 ' update control states If colC.Count > 0 Then

UpdateTab1ControlStates

End If

End Sub

'accepts a string and parses it for undefined variable names. Returns a 'collection of the variable names that are unique.

Public Function UndefinedNames(ByVal strS As String) As Collection

```
Dim lngStart As Long
 5
          Dim lngEnd As Long
          Dim strT As String
          Dim byt1 As Byte
          Dim byt2 As Byte
          Dim colC As New Collection
10
          Dim blnDup As Boolean
          Dim varT As Variant
          ' parse the variable names out of strS
          For lngStart = 1 To Len(strS)
             byt1 = Asc(Mid(strS, lngStart, 1))
             If byt1 \geq 65 And byt1 \leq 90 Then
               For lngEnd = lngStart + 1 To Len(strS)
                 byt2 = Asc(Mid(strS, lngEnd, 1))
                 Select Case byt2
                    Case 48 To 57, 65 To 90, 97 To 122
                      ' if it's 0 to 9, A to Z, or a to z, continue searching
                    Case Else
                      ' if it's not, assume end of variable name has been found
                      Exit For
                 End Select
               Next lngEnd
               strT = Mid(strS, lngStart, lngEnd - lngStart)
               ' throw name away if it's already in colC
               blnDup = False
               For Each varT In colC
30
                 If UCase(varT) = UCase(strT) Then
                    blnDup = True
                 End If
               Next varT
               ' make sure name is not a Prolog function
35
               If blnDup = False Then
                 ' throw name away if it's already in the main variable collection
                 If mudtFam.ActiveModel.Variables.UniqueName(strT) Then
                    Call colC.Add(strT)
40
                 End If
```

End If

```
lngStart = lngEnd
           End If
         Next IngStart
         Set UndefinedNames = colC
       End Function
       Private Sub TestConstraints(ByVal udtTestType As TestType)
         Dim strVN As String
         Dim blnUnderconstrained As Boolean
         Dim blnTestAborted As Boolean
         If mudtFam.ActiveModel.ConstraintsOK(udtTestType, mudtProlog,
           blnUnderconstrained, blnTestAborted, strVN) Then
              Call MsgBox("Looks good!", vbExclamation, "Test Result")
         ElseIf blnTestAborted Then
           Call MsgBox("Test aborted!", vbExclamation, "Test Result")
         ElseIf blnUnderconstrained Then
           Call MsgBox("Variable " & strVN & " is underconstrained!", _
G1
U1
              vbExclamation, "Test Result")
         Else
20
           Call MsgBox("No solutions exist!", vbExclamation, "Test Result")
         End If
       End Sub
C)
       ' displays the family attributes on the status bar
       Private Sub UpdateFamilyAttributes()
         Select Case mudtFam.Program
           Case prGRE
              stbS.Panels(pnProgramName) = "GRE"
           Case prGMAT
              stbS.Panels(pnProgramName) = "GMAT"
           Case prSAT
              stbS.Panels(pnProgramName) = "SAT"
         End Select
         Select Case mudtFam.ItemType
           Case ptStandardMC
              stbS.Panels(pnItemType) = "SMC"
           Case ptQuantComp
```

5

10

30

35

```
stbS.Panels(pnItemType) = "QC"
             Case ptDataSuff
               stbS.Panels(pnItemType) = "DS"
          End Select
 5
          If mudtFam.Generic Then
             stbS.Panels(pnGeneric) = "Generic"
          Else
             stbS.Panels(pnGeneric) = "Non generic"
          End If
10
          Select Case mudtFam.Proximity
             Case prNear
               stbS.Panels(pnProximity) = "Near"
             Case prMedium
               stbS.Panels(pnProximity) = "Medium"
             Case prFar
15
               stbS.Panels(pnProximity) = "Far"
          End Select
        End Sub
Mr. of the Con Line
        ' returns the model file name given the doc file name
2∯
        Private Function ModelFileName(ByVal strDocFN As String) As String
 J)
          ModelFileName = left(strDocFN, Len(strDocFN) - 4) & ".mdl"
 ď)
        End Function
 j=h
        'extracts the key from a model file name
        Private Function ModelKey(ByVal strFN As String) As String
25
          Dim varI1 As Variant
          Dim varI2 As Variant
          Dim intI As Integer
          Dim strS As String
          varI1 = InStr(1, strFN, "$")
          varI2 = InStr(varI1, strFN, ".")
30
          ' strip off numbers or spaces to the left of the "."
          intI = varI2
          Do While intI > varI1
             intI = intI - 1
```

```
strS = Mid(strFN, intI, 1)
             If Asc(strS) >= 65 And Asc(strS) <= 91 Then 'it's A to Z
               varI2 = intI + 1
               Exit Do
             End If
 5
          Loop
          ModelKey = Mid(strFN, varI1 + 1, varI2 - varI1 - 1)
        End Function
        'embeds a new key into a model file name
        Private Function ModelEmbedKey(ByVal strFN As String, ByVal strNewKey As String) _
10
          As String
          Dim varI1 As Variant
          Dim varI2 As Variant
          Dim intl As Integer
          Dim strS As String
          varI1 = InStr(1, strFN, "$")
<u>O</u>1
The Was design
          varI2 = InStr(varI1, strFN, ".")
          ' strip off numbers or spaces to the left of the "."
          intI = varI2
20
          Do While intI > varI1
             intI = intI - 1
             strS = Mid(strFN, intI, 1)
             If Asc(strS) \ge 65 And Asc(strS) \le 91 Then 'it's A to Z
               varI2 = intI + 1
               Exit Do
             End If
          Loop
          ModelEmbedKey = left(strFN, varI1) & strNewKey & right(strFN, 4)
        End Function
        ' returns the key of the next child for this model
30
        Private Function NextModelKey(strFN As String) As String
          Dim nodN As Node
          Dim strNewFN As String
          Dim strIndex As String
35
          Dim strT As String
```

```
Dim intl As Integer
          ' when the key can't be found in the Nodes collection, an error
          ' is raised. When the error is raised, the first available letter
          ' of the alphabet has been found.
 5
          On Error GoTo Found
          For intI = 65 \text{ To } 90 \text{ '} \text{ A thru } Z
             strT = Chr(intI)
             Set nodN = treModels.Nodes.Item(strIndex & strT)
10
          Next intI
          On Error GoTo 0
          Call MsgBox("Can't add another child model to this parent", _
             vbExclamation, "Error")
          Exit Function
 Jî.
 U1
1$
        Found:
          NextModelKey = strIndex & strT
          Exit Function
        End Function
        ' resets controls and variables when a new family is opened.
        Private Sub ClearControls()
          If mudtFam Is Nothing Then
             ' do nothing
          Else
             mudtFam.WriteFamily
25
               If mudtFam.ActiveModel Is Nothing Then
                  ' do nothing
                Else
                  mudtFam.ActiveModel.WriteModel
               End If
30
          End If
          mudtWord.CloseAllDocs
```

strIndex = ModelKey(strFN)

	Set mudtFam = Nothing
	Set mudtClone = Nothing
	treModels.Nodes.Clear
	lstVariables.Clear
5	lstDisposition.Clear
J	lstAccepted.Clear
	stbS.Panels(pnProgramName) = ""
	stbS.Panels(pnFamilyName) = ""
	stbS.Panels(pnItemType) = ""
10	stbS.Panels(pnGeneric) = ""
10	stbS.Panels(pnProximity) = ""
	stbS.Panels(pnActiveModelIcon).Picture = Nothing
	stbS.Panels(pnActiveModelName) = ""
	frmComments.Comment = ""
15	mnuAcceptedCopy.Enabled = False
13	mnuAcceptedCopy.Enabled = False
	milar recepted aste. Endoted Taise
Fa	End Sub
199 199	
<u> </u>	' used to reformat tab 2 as QC and DS don't need a distractor listbox
ű	Private Sub FormatTab2(ByVal udtItemType As ItemType)
201	Select Case udtItemType
20 m	Case ptStandardMC
43	' turn on the distractor list box
£ p=q	lblDistractor.Visible = True
111 111	lstConstraints(1).Visible = True
2 5	cmdConstraintAdd(1). Visible = True
jes jes	cmdConstraintEdit(1).Visible = True
en En	cmdConstraintRemove(1).Visible = True
	cmdConstraintTest(1).Visible = True
	Case ptQuantComp
30	'turn off the distractor list box
	lblDistractor.Visible = False
	lstConstraints(1).Visible = False
	cmdConstraintAdd(1).Visible = False
	cmdConstraintEdit(1).Visible = False
35	cmdConstraintRemove(1).Visible = False
	cmdConstraintTest(1).Visible = False
	Case ptDataSuff
	' turn off the distractor list box
40	lblDistractor.Visible = False
40	lstConstraints(1).Visible = False
	cmdConstraintAdd(1).Visible = False

```
cmdConstraintEdit(1).Visible = False
               cmdConstraintRemove(1).Visible = False
               cmdConstraintTest(1).Visible = False
          End Select
 5
        End Sub
        'this method gets rid of all variants in the lstDisposition listbox,
        'deletes them from disk, and removes them from the active model.
        Private Sub KillVariants()
          Dim udtClone As Clone
10
          Dim intl As Integer
          With lstDisposition
            For intI = 0 To .ListCount - 1
               get object from active model's clone collection
               Set udtClone = mudtFam.ActiveModel.Clones.Item(Str(.ItemData(intI)))
               ' close the document
               udtClone.CloseDoc
               ' delete the clone file
               Kill IN DIRECTORY & udtClone.FileName
               ' remove the clone from the active model's collection
               Call mudtFam.ActiveModel.Clones.Remove(Str(.ItemData(intI)))
            Next intI
            For intI = .ListCount - 1 To 0 Step -1
               'remove the entry from the disposition list box
               Call .RemoveItem(intI)
            Next intI
          End With
        End Sub
       Private Sub UpdateTab0ControlStates()
          ' update model tree menu states
30
          With treModels
            If .Nodes.Count > 0 Then
               mnuTreeExtend.Enabled = True
               mnuTreeRemove.Enabled = True
               cmdTreeExtend.Enabled = True
               cmdTreeRemove.Enabled = True
35
            Else
               mnuTreeExtend.Enabled = False
```

	mnu i reeRemove. Enabled = raise
	cmdTreeExtend.Enabled = False
	cmdTreeRemove.Enabled = False
	End If
5	End With
3	
	' update accepted list box menu states
	With lstAccepted
	If .ListCount > 0 Then
	cmdPrintBatch.Enabled = True
10	If .SelCount = 1 Then ' 1 item is selected
10	
	mnuAcceptedProfile.Enabled = True
	mnuAcceptedCopy.Enabled = True
	cmdAcceptedEdit.Enabled = True
	cmdAcceptedCopy.Enabled = True
15	ElseIf .SelCount > 1 Then ' more than one is selected
	mnuAcceptedProfile.Enabled = False
	mnuAcceptedCopy.Enabled = False
F= 4	cmdAcceptedEdit.Enabled = False
≒# .}1	cmdAcceptedCopy.Enabled = False
2€	End If
20 5 February 25	Else 'nothings in the list box
= : = :	cmdPrintBatch.Enabled = False
17	mnuAcceptedProfile.Enabled = False
mje Em	mnuAcceptedCopy.Enabled = False
25	mnuAcceptedPaste.Enabled = False
	cmdAcceptedEdit.Enabled = False
	cmdAcceptedCopy.Enabled = False
41	cmdAcceptedPaste.Enabled = False
===	End If
2 <u>h</u>	End With
	Elia Willi
	If mudtClane Is Nothing Then ! nothing to necto
	If mudtClone Is Nothing Then 'nothing to paste
	mnuAcceptedPaste.Enabled = False
	cmdAcceptedPaste.Enabled = False
2.5	ElseIf lstAccepted.SelCount > 0 Then 'one or more are selected
35	mnuAcceptedPaste.Enabled = True
	cmdAcceptedPaste.Enabled = True
	Else 'none are selected
	mnuAcceptedPaste.Enabled = False
	cmdAcceptedPaste.Enabled = False
40	End If
	If mudtFam Is Nothing Then
	cmdDone.Enabled = False

```
Else
            cmdDone.Enabled = True
         End If
       End Sub
 5
       Private Sub UpdateTab1ControlStates(Optional ByVal intIndex As Integer = 0)
         Dim strCaption As String
         If mudtFam.ActiveModel.IsFrozen Then
            strCaption = "Browse"
         Else
10
            strCaption = "Edit"
          End If
          mnuVariablesEdit.Caption = strCaption
          cmdVariableEdit.Caption = strCaption
          mnuConstraintsEdit.Caption = strCaption
15
          cmdConstraintEdit(0).Caption = strCaption
          cmdConstraintEdit(1).Caption = strCaption
          ' update variable list box menu states
          If mudtFam.ActiveModel.IsFrozen Then
            mnuVariablesAdd.Enabled = False
            mnuVariablesEdit.Enabled = True
            mnuVariablesEnableAll.Enabled = False
            mnuVariablesDisableAll.Enabled = False
            mnuVariablesRemove.Enabled = False
            mnuVariablesRemoveAll.Enabled = False
            cmdVariableAdd.Enabled = False
            cmdVariableEdit.Enabled = True
            cmdVariableRemove.Enabled = False
          ElseIf lstVariables.ListCount > 0 Then
            mnuVariablesAdd.Enabled = True
            mnuVariablesEdit.Enabled = True
30
            mnuVariablesEnableAll.Enabled = True
            mnuVariablesDisableAll.Enabled = True
            mnuVariablesRemove.Enabled = True
            mnuVariablesRemoveAll.Enabled = True
            cmdVariableAdd.Enabled = True
35
            cmdVariableEdit.Enabled = True
            cmdVariableRemove.Enabled = True
          Else
            mnuVariablesAdd.Enabled = True
```

5	mnuVariablesEdit.Enabled = False mnuVariablesEnableAll.Enabled = False mnuVariablesDisableAll.Enabled = False mnuVariablesRemove.Enabled = False mnuVariablesRemoveAll.Enabled = False cmdVariableAdd.Enabled = True cmdVariableEdit.Enabled = False cmdVariableRemove.Enabled = False End If
10	' isfrozen should not effect state of test option If lstVariables.ListCount > 0 Then mnuVariablesTest.Enabled = True cmdVariableTest.Enabled = True
15	Else mnuVariablesTest.Enabled = False cmdVariableTest.Enabled = False End If
200 mp man con a con	' update constraints list box menu states If mudtFam. ActiveModel. IsFrozen Then mnuConstraintsAdd. Enabled = False mnuConstraintsEdit. Enabled = True mnuConstraintsEnableAll. Enabled = False mnuConstraintsDisableAll. Enabled = False mnuConstraintsRemove. Enabled = False mnuConstraintsRemoveAll. Enabled = False cmdConstraintAdd(0). Enabled = False cmdConstraintAdd(1). Enabled = False cmdConstraintEdit(0). Enabled = True cmdConstraintEdit(1). Enabled = True cmdConstraintRemove(0). Enabled = False cmdConstraintRemove(1). Enabled = False ElseIf lstConstraints(intIndex). ListCount > 0 Then mnuConstraintsAdd. Enabled = True
35	mnuConstraintsEdit.Enabled = True mnuConstraintsEnableAll.Enabled = True mnuConstraintsDisableAll.Enabled = True mnuConstraintsRemove.Enabled = True mnuConstraintsRemoveAll.Enabled = True
40	<pre>cmdConstraintAdd(intIndex).Enabled = True cmdConstraintEdit(intIndex).Enabled = True cmdConstraintRemove(intIndex).Enabled = True Flse</pre>

mnuConstraintsAdd.Enabled = True

5	mnuConstraintsEdit.Enabled = False mnuConstraintsEnableAll.Enabled = False mnuConstraintsDisableAll.Enabled = False mnuConstraintsRemove.Enabled = False mnuConstraintsRemoveAll.Enabled = False cmdConstraintAdd(intIndex).Enabled = True cmdConstraintEdit(intIndex).Enabled = False cmdConstraintRemove(intIndex).Enabled = False End If
10	' isfrozen should not effect state of test option If lstConstraints(intIndex).ListCount > 0 Then mnuConstraintsTest.Enabled = True cmdConstraintTest(intIndex).Enabled = True Else
15	mnuConstraintsTest.Enabled = False cmdConstraintTest(intIndex).Enabled = False End If
1.1. 1.1. 1.1. 1.1. 1.1. 1.1. 1.1. 1.1	' flip the index If intIndex = 0 Then intIndex = 1 Else intIndex = 0 End If
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1	' update button states for the other constraint list box If mudtFam.ActiveModel.IsFrozen = False Then If lstConstraints(intIndex).ListCount > 0 Then cmdConstraintAdd(intIndex).Enabled = True cmdConstraintEdit(intIndex).Enabled = True cmdConstraintRemove(intIndex).Enabled = True Else cmdConstraintAdd(intIndex).Enabled = True cmdConstraintEdit(intIndex).Enabled = False cmdConstraintRemove(intIndex).Enabled = False cmdConstraintRemove(intIndex).Enabled = False
35	End If
	' isfrozen should not effect state of test option If lstConstraints(intIndex).ListCount > 0 Then cmdConstraintTest(intIndex).Enabled = True Else
40	<pre>cmdConstraintTest(intIndex).Enabled = False End If</pre>

	' update import button If mudtFam.ActiveModel.IsFrozen Then amdImportConstraints Enabled = False
	cmdImportConstraints.Enabled = False Else
5	cmdImportConstraints.Enabled = True
,	End If
	' if model frozen, disable save
	If mudtFam. ActiveModel.IsFrozen Then
	cmdSaveModel.Enabled = False
10	Else
	If mudtFam.ActiveModel.IsDirty Then
	cmdSaveModel.Enabled = True
	Else cmdSaveModel.Enabled = False
15	End If
13	End If
	Liid II
f=3	End Sub
() " (
۵ì	Private Sub UpdateTab2ControlStates()
L	
	'update disposition list box menu states
2 U j	If lstDisposition.ListCount > 0 And cmdGenerate.Caption = "Generate" Then
-d= .{1	mnuDispAccept.Enabled = True mnuDispDefer.Enabled = True
===	mnuDispDeter.Enabled = True
C.	mnuDispMakeModel.Enabled = True
25	cmdPrintVariants.Enabled = True
	cmdPrintVariants.Enabled = True
⊨k ea	cmdDispAccept.Enabled = True
ind party	cmdDispDefer.Enabled = True
122 E	cmdDispDiscard.Enabled = True
30	cmdDispMakeModel.Enabled = True
	Else
	mnuDispAccept.Enabled = False
	mnuDispDefer.Enabled = False
2.5	mnuDispDiscard.Enabled = False
35	mnuDispMakeModel.Enabled = False
	cmdPrintVariants.Enabled = False
	cmdPrintVariants.Enabled = False cmdDispAccept.Enabled = False
	cmdDispAccept.Enabled = False cmdDispDefer.Enabled = False
40	cmdDispDeter.Enabled = False
	cmdDispMakeModel Fnabled = False

VBSCA -216-

```
' Variable.frm
       VERSION 5.00
       Object = "{6B7E6392-850A-101B-AFC0-4210102A8DA7}#1.3#0"; "COMCTL32.OCX"
       Object = "{F9043C88-F6F2-101A-A3C9-08002B2F49FB}#1.2#0"; "COMDLG32.OCX"
 5
       Begin VB.Form frmVariable
        BorderStyle
                     = 4 'Fixed ToolWindow
                   = "Create or Change Variable"
        Caption
        ClientHeight = 4230
        ClientLeft
                    = 45
        ClientTop
                    = 285
10
        ClientWidth = 6525
                    = "Form1"
        LinkTopic
        MaxButton
                     = 0 'False
                     = 0 'False
        MinButton
        ScaleHeight = 4230
15
        ScaleWidth
                     = 6525
        ShowInTaskbar = 0 'False
        StartUpPosition = 1 'CenterOwner
        Begin VB.ComboBox cboVarType
          Height
                     = 315
20
          ItemData
                     = "Variable.frx":0000
          Left
                   = 2040
          List
                   = "Variable.frx":0013
          Style
                    = 2 'Dropdown List
25
          TabIndex
          ToolTipText = "Select the variable type."
          Top
                    = 360
          Width
                     = 1695
        End
        Begin VB.CheckBox chkChecksum
                     = "Add to checksum"
          Caption
          Height
                    = 375
          Left
                   = 240
          TabIndex
                       = "Check this box to add this variable to the checksum calcuation."
35
          ToolTipText
          Top
                    = 840
          Value
                    = 1 'Checked
          Width
                    = 1815
        End
40
        Begin MSComDlg.CommonDialog cdlCD
                   = 5280
          Left
          Top
                    = 2520
          ExtentX
                      = 847
          ExtentY
                      = 847
```

```
Version
                      = 393216
         End
         Begin VB.CommandButton cmdVarExport
                      = "Export Strings"
          Height
                      = 495
 5
          Left
                    = 5160
          TabIndex
                       = 7
          ToolTipText = "Click here to export a set of strings."
          Top
                     = 1920
10
          Width
                      = 1215
         End
         Begin VB.CommandButton cmdVarImport
                      = "Import Strings"
          Caption
          Height
                      = 495
15
          Left
                    = 5160
                       = 6
          TabIndex
          ToolTipText = "Click here to import a set of strings."
          Top
                     = 1320
          Width
                      = 1215
20j
         End
         Begin VB.TextBox txtVariableName
 đ1
Mer. May Com
          Height
                     = 315
          Left
                    = 240
          TabIndex
                       = 0
2€=
          ToolTipText = "Enter the variable name here."
 ű
          Top
                     = 360
          Width
                     = 1695
         End
         Begin VB.CommandButton cmdVarCancel
          Caption
                      = "Cancel"
          Height
                      = 495
          Left
                    = 5160
          TabIndex
                       = 5
          ToolTipText = "Click here to return without saving changes."
35
                     = 720
          Top
          Width
                      = 1215
         End
         Begin VB.CommandButton cmdVarOK
          Caption
                      = "OK"
40
          Default
                      = -1 'True
                     = 495
          Height
          Left
                    = 5160
          TabIndex
                       = 4
          ToolTipText = "Click here to save changes and return."
45
                     = 120
```

Top

```
Width
                     = 1215
        End
        Begin ComctlLib.ListView lvwTemp
          Height
                    = 375
                   = 5280
5
          Left
          TabIndex
                      = 43
                    = 3120
          Top
                    = 0 'False
          Visible
          Width
                     = 495
                      = 873
10
          ExtentX
          ExtentY
                      = 661
          View
                    = 3
                     = 2
          Arrange
                     = 1
          LabelEdit
15
          MultiSelect = -1 'True
                       = -1 'True
          LabelWrap
          HideSelection = -1 'True
           Version
                     = 327682
          ForeColor
                      = -2147483640
                      = -2147483643
          BackColor
          BorderStyle
                      = 1
g
M. 444 A.
          Appearance
                       = 1
          NumItems
                       = 0
        End
25-
        Begin ComctlLib.ListView lvwDummy
ij,
          Height
                     = 375
          Left
                   = 5280
30
          TabIndex
                      = 44
                    = 3600
          Top
          Visible
                    = 0 'False
          Width
                     = 495
          _ExtentX
                      = 873
                      = 661
           ExtentY
          View
                    = 3
                     = 2
35
          Arrange
          LabelEdit
                     = 1
          MultiSelect = -1 'True
                       = -1 'True
          LabelWrap
          HideSelection = -1 'True
                      = 327682
40
           Version
          ForeColor
                      = -2147483640
          BackColor
                      = -2147483643
          BorderStyle
                      = 1
          Appearance
                       = 1
45
          NumItems
                       = 0
```

```
End
        Begin VB.Frame fraString
          BorderStyle = 0 'None
                     = 2895
          Height
5
          Left
                    = 240
          TabIndex
                      = 9
                    = 1200
          Top
                     = 4815
          Width
          Begin ComctlLib.ListView lvwStrings
10
           Height
                       = 1815
           Left
                     = 0
           TabIndex
                        = 42
            Top
                      = 720
            Width
                       = 3975
15
                        = 7011
            ExtentX
            ExtentY
                        = 3201
            View
                       = 3
                       = 2
            Arrange
           LabelEdit
                        = 1
           MultiSelect = -1 'True
                        = -1 'True
           LabelWrap
Mr. An Com
           HideSelection = -1 'True
            Version
                       = 327682
           ForeColor
                        = -2147483640
25=
           BackColor
                        = -2147483643
ű
           BorderStyle
                        = 1
                         = 1
           Appearance
           NumItems
                         = 0
          End
          Begin VB.CheckBox chkIndexed
                       = "Indexed"
           Caption
           Height
                       = 375
           Left
                     = 0
                        = 41
           TabIndex
           ToolTipText = "Check this box for indexed strings."
35
           Top
            Width
                       = 1215
          Begin VB.CommandButton cmdRemove
40
                       = "Remove"
           Caption
                       = 255
           Height
           Left
                     = 2640
           TabIndex
                        = 40
                        = "Click here to remove a set of indexed values."
           ToolTipText
                      = 2520
```

Top

```
Width
                      = 1335
         · End
          Begin VB.CommandButton cmdEdit
                       = "Edit"
           Caption
 5
           Height
                      = 255
           Left
                     = 1320
                        = 39
           TabIndex
           ToolTipText = "Click here to edit a set of indexed values."
                      = 2520
           Top
10
           Width
                      = 1335
          End
          Begin VB.CommandButton cmdAdd
                       = "Add"
           Caption
                      = 255
           Height
           Left
                     = 0
15
           TabIndex
                        = 38
           ToolTipText = "Click here to add a new set of indexed values."
           Top
                      = 2520
            Width
                      = 1335
          End
          Begin VB.Label lblStringVals
           Caption
                       = "String values"
           Height
                      = 255
           Left
                     = 0
25
                      = 37
           TabIndex
M)
                      = 480
           Top
            Width
                      = 1695
          End
        End
        Begin VB.Frame fraUntyped
          BorderStyle = 0 'None
          Height
                     = 2895
          Left
                    = 240
          TabIndex
                      = 35
35
          Top
                    = 1200
          Width
                     = 4815
          Begin VB.TextBox txtUntyped
           Height
                      = 2295
           Left
                     = 240
                       = -1 'True
40
           Locked
                        = -1 'True
           MultiLine
           TabIndex
                        = 36
           ToolTipText = "Interesting, no?"
           Top
                      = 360
45
            Width
                      = 4335
```

```
End
         End
         Begin VB.Frame fraIndependent
          BorderStyle = 0 'None
                      = "Frame1"
 5
          Caption
          Height
                     = 2895
          Left
                    = 240
                       = 10
          TabIndex
          Top
                     = 1200
          Width
                     = 4815
10
          Begin VB.CheckBox chkIsIndependent
            Caption
                       = "Independent"
            Height
                       = 375
            Left
                      = 0
15
            TabIndex
                        = 11
                         = "Check this box if the value of this variable is not dependent."
            ToolTipText
                      = 0
            Top
                       = 1 'Checked
            Value
            Width
                       = 1575
2Q;
          End
          Begin VB.Frame fraRealFormat
 ٥ì
 L.
            BorderStyle = 0 'None
            Height
                       = 1095
 43
            Left
                      = 0
25
            TabIndex
                        = 26
 ď)
            Top
                      = 1680
            Width
                       = 4815
 Begin VB.CheckBox chkOnGrid
                         = "Value must be multiple of precision"
             Caption
             Height
                         = 375
             Left
                       = 1800
 Ej
             TabIndex
                          = 45
                        = 120
             Top
              Width
                         = 2895
35
            End
            Begin VB.ComboBox cboPrecision
                         = 315
             Height
             ItemData
                          = "Variable.frx":0041
             Left
                       = 120
40
             List
                       = "Variable.frx":0060
             Style
                        = 2 'Dropdown List
             TabIndex
                          = 34
                        = 360
             Top
             Width
                         = 1455
```

End

```
Begin VB.CheckBox chkTrailingZeros
             Caption
                         = "Display trailing zeros"
             Height
                        = 375
             Left
                       = 1800
 5
             TabIndex
                          = 28
             Top
                        = 480
             Width
                        = 1935
            End
            Begin VB.Label LblDecimals
                         = "Precision"
10
             Caption
             Height
                        = 255
             Left
                       = 480
             TabIndex
                          = 29
                        = 120
             Top
15
             Width
                        = 1095
           End
          End
          Begin VB.Frame fraFractionFormat
           BorderStyle = 0 'None
           Caption
                       = "Frame1"
           Height
                       = 1215
 ٥٦
 Mr. den den
           Left
                     = -120
                        = 32
           TabIndex
                      = 1560
           Top
                       = 5055
            Width
 ű,
           Begin VB.CheckBox chkMixedNumbers
                         = "Mixed numbers"
             Caption
             Height
                        = 375
             Left
                       = 1560
             TabIndex
                         = 33
             ToolTipText
                           = "Check this box if you wish improper fractions to be converted into
       mixed numbers."
             Top
                        = 240
             Width
                        = 1695
35
           End
          End
          Begin VB.Frame fraIntRealRange
           BorderStyle = 0 'None
           Height
                       = 1335
40
           Left
                      = 0
           TabIndex
                        = 22
           Top
                      = 360
            Width
                       = 4815
           Begin VB.TextBox txtBy
45
             Height
                        = 315
```

```
Left
                        = 3240
             TabIndex
                          = 25
                          "1"
             Text
                           = "Enter the increment here. Variables and expressions may be used."
             ToolTipText
 5
             Top
                        = 600
             Width
                         = 1455
            End
            Begin VB.TextBox txtTo
             Height
                         = 315
                        = 1680
10
             Left
             TabIndex
                          = 24
                        = "100"
             Text
                           = "Enter the value in the range here. Variables and expressions may be
       used."
15
             Top
                        = 600
                         = 1455
             Width
            End
            Begin VB.TextBox txtFrom
             Height
                         = 315
2Q)
             Left
                        = 120
             TabIndex
                          = 23
 D
 Ü
                        = "1"
             Text
 ToolTipText = "Enter the lowest value in the range here. Variables and expressions
       may be used."
             Top
                        = 600
             Width
                         = 1455
            End
 Begin VB.Label lblBy
 ű,
                         = "By"
             Caption
3<u>0</u>
             Height
                         = 255
             Index
                        = 0
             Left
                        = 3840
             TabIndex
                          = 31
             Top
                        = 360
             Width
                         = 495
35
            End
            Begin VB.Label lblTo
             Caption
                         = "To"
             Height
                         = 255
40
             Index
                        = 0
             Left
                        = 2280
             TabIndex
                          = 30
             Top
                        = 360
             Width
                         = 615
```

End

```
Begin VB.Label lblFrom
             Caption
                         = "From"
             Height
                        = 255
             Index
                        = 0
                       = 720
 5
             Left
             TabIndex
                         = 27
                        = 360
             Top
             Width
                        = 975
            End
10
          End
          Begin VB.Frame fraFractionRange
            BorderStyle = 0 'None
            Height
                       = 1455
                     = 0
            Left
15
            TabIndex
                        = 12
            Top
                      = 360
                       = 4815
            Width
            Begin VB.TextBox txtByNum
             Height
                        = 315
 C
20
             Left
                       = 3240
 <u>D</u>1
             TabIndex
                          = 18
 = "1"
             Text
             ToolTipText = "Enter the numerator of the increment here."
             Top
                        = 360
             Width
                        = 1455
            End
            Begin VB.TextBox txtToNum
             Height
                        = 315
IJ,
             Left
                       = 1680
 TabIndex
                          = 17
30.
                        = "100"
Text
             ToolTipText = "Enter the numerator of the highest value in the range here."
             Top
                        = 360
             Width
                        = 1455
35
            End
            Begin VB.TextBox txtFromNum
             Height
                        = 315
             Left
                       = 120
             TabIndex
                          = 16
40
             Text
                       = "1"
             ToolTipText = "Enter the numerator of the lowest value of the range here."
                        = 360
             Top
             Width
                        = 1455
            End
```

Begin VB.TextBox txtFromDen

45

```
Height
                         = 315
              Left
                        = 120
              TabIndex
                           = 15
                        = "1"
              Text
 5
              ToolTipText = "Enter the denominator of the lowest value in the range here."
              Top
                        = 840
              Width
                         = 1455
            End
            Begin VB.TextBox txtToDen
10
              Height
                         = 315
              Left
                        = 1680
              TabIndex
                           = 14
                        = "1"
              Text
              ToolTipText = "Enter the denominator of the highest value in the range here."
15
              Top
                        = 840
              Width
                         = 1455
            End
            Begin VB.TextBox txtByDen
 Height
                         = 315
201
              Left
                        = 3240
              TabIndex
                           = 13
                        = "1"
              Text
                           = "Enter the denominator of the increment here."
              ToolTipText
              Top
                        = 840
              Width
                         = 1455
            End
 17 45, 17 16 in
            Begin VB.Label lblBy
              Caption
                         = "By"
              Height
                         = 255
304
              Index
                         = 1
 C)
              Left
                        = 3840
 C)
              TabIndex
                          = 21
                        = 120
              Top
              Width
                         = 255
35
            End
            Begin VB.Label lblTo
              Caption
                         = "To"
              Height
                         = 255
              Index
                         = 1
40
              Left
                        = 2280
                          = 20
              TabIndex
              Top
                        = 120
              Width
                         = 375
            End
```

Begin VB.Label lblFrom

```
= "From"
             Caption
             Height
                       = 255
             Index
                       = 1
             Left
                      = 480
 5
             TabIndex
                         = 19
             Top
                       = 120
             Width
                        = 495
           End
           Begin VB.Line Line1
10
             BorderWidth
             Index
                       = 0
             X1
                       = 120
             X2
                       = 1560
             Y1
                       = 750
15
             Y2
                       = 750
           End
           Begin VB.Line Line1
             BorderWidth
                          = 3
                       = 1
             Index
201
             X1
                       = 1680
25%
             X2
                       = 3120
                       = 750
             Y1
             Y2
                       = 750
           End
           Begin VB.Line Line1
             BorderWidth
                          = 3
3Q.
             Index
                       = 2
             X1
                       = 3240
             X2
                       = 4680
             Y1
                       = 750
                       = 750
             Y2
           End
         End
        End
35
        Begin VB.Label lblVarType
          Caption
                     = "Type"
         Height
                    = 255
          Left
                   = 2040
          TabIndex
                      = 8
40
          Top
                    = 120
          Width
                    = 1095
        End
        Begin VB.Label lblVarName
          Caption
                     = "Variable Name"
```

Height

= 255

	Left $= 240$
	TabIndex = 3
	Top = 120
	Width $= 1095$
5	End
	Begin VB.Menu mnuString
	Caption = "String"
	Visible = 0 'False
	Begin VB.Menu mnuStringAdd
10	Caption = "Add"
	End
	Begin VB.Menu mnuStringEdit
	Caption = "Edit"
	End
15	Begin VB.Menu mnuStringRemove
	Caption = "Remove"
	End
	End
	End
20]	Attribute VB_Name = "frmVariable"
g1	Attribute VB_GlobalNameSpace = False
Lil	Attribute VB_Creatable = False
25.00 to the state of the state	Attribute VB_PredeclaredId = True
#3	Attribute VB_Exposed = False
25.	Option Explicit
	Private mudtVar As Variable
ű	Private mudtVarInt As VarInteger
	Private mudtVarReal As VarReal
<u>L</u> L	Private mudtVarFraction As VarFraction
3 0	Private mudtVarString As VarString
2	Private mudtVarUntyped As VarUntyped
	' to see if the variable type has changed
	Private mudtType As VariableType
	Private mudtOldType As VariableType
35	'needed for string list box
	Private mbytAddEditFlag As Byte
	'needed for listbox update
	Private mlstListBox As ListBox
	'current active model
40	Private mudtModel As Model

VBSCA -228-

Public Property Let AddEditFlag(ByVal bytNewValue As Byte) mbytAddEditFlag = bytNewValue End Property Public Property Get AddEditFlag() As Byte AddEditFlag = mbytAddEditFlag **End Property** Public Property Let Variable(ByVal udtNewValue As Variable) Set mudtVar = udtNewValue 10 **End Property** Public Property Let ListBox(ByVal lstNewValue As ListBox) 43 M Set mlstListBox = lstNewValue Min Mar Die Man Con **End Property** Public Property Let Model(ByVal udtNewValue As Model) 1**5** Set mudtModel = udtNewValue **End Property** Private Sub chkIndexed Click() Call CopyListView(lvwStrings, lvwTemp) Call CopyListView(lvwDummy, lvwStrings) Call CopyListView(lvwTemp, lvwDummy) 20 End Sub Private Sub CopyListView(ByVal lvw1 As ListView, lvw2 As ListView) Dim intl As Integer Dim intI2 As Integer Dim lsiItem As ListItem 25 'copy visible listview into temp listview

```
lvw2.ListItems.Clear
          lvw2.ColumnHeaders.Clear
          For intI = 1 To lvw1.ColumnHeaders.Count
            Call lvw2.ColumnHeaders.Add(,, lvw1.ColumnHeaders(intI))
 5
          Next intI
          For intI = 1 To lvw1.ListItems.Count
            Set lsiItem = lvw2.ListItems.Add(, , lvw1.ListItems.Item(intI).Text)
            For intI2 = 1 To lvw1.ColumnHeaders.Count - 1
10
               lsiItem.SubItems(intI2) = lvw1.ListItems.Item(intI).SubItems(intI2)
            Next intI2
          Next intI
15
       End Sub
       Private Sub cmdAdd Click()
          Call mnuStringAdd Click
 T)
 ō
U
       End Sub
20
       Private Sub cmdEdit Click()
          Call mnuStringEdit_Click
       End Sub
 IJ,
       Private Sub cmdRemove Click()
 Ĉ
 Lik
 C)
          Call mnuStringRemove Click
 C)
25
       End Sub
       Private Sub Form_Load()
          Dim udtWAPI As New Win32API
          ' enable full row select
30
          Call udtWAPI.EnableListViewFullRowSelect(lvwStrings)
          ' load up explanation of untyped variables
          txtUntyped = LoadResString(1)
          cboVarDelimiter.ListIndex = 0 ' default to "@"
```

```
cboPrecision.ListIndex = 1 ' default to ".01"
          cdlCD.CancelError = True
          If mbytAddEditFlag = aeEdit Then
 5
            txtVariableName = mudtVar.name
            If mudtVar.Checksum Then
               chkChecksum = 1
10
            Else
               chkChecksum = 0
            End If
            Select Case TypeName(mudtVar)
15
               Case "VarInteger"
                 Set mudtVarInt = mudtVar
                 With mudtVarInt
                   txtFrom = .From
 41
201
                   txtTo = .Too
25 44 44 44
                   txtBy = .By
                   If .IsIndependent Then
                     chkIsIndependent = 1
                   Else
                     chkIsIndependent = 0
                   End If
                 End With
                 mudtType = vtInteger
<u>L</u>£
              Case "VarReal"
30
                 Set mudtVarReal = mudtVar
IJ
                 With mudtVarReal
                   txtFrom = .From
                   txtTo = .Too
35
                   txtBy = .By
                   If .IsIndependent Then
                     chkIsIndependent = 1
                   Else
                      chkIsIndependent = 0
40
                   End If
                   If .IsOnGrid Then
                      chkOnGrid = 1
                   Else
                     chkOnGrid = 0
```

	End If
	If .TrailingZeros Then
	chkTrailingZeros = 1
	Else
5	chkTrailingZeros = 0
	End If
	cboPrecision = .Precision
	End With
	mudtType = vtReal
10	•
	Case "VarFraction"
	Set mudtVarFraction = mudtVar
	With mudtVarFraction
	txtFromNum = .FromNumerator
15	txtFromDen = .FromDenominator
	txtToNum = .ToNumerator
	txtToDen = .ToDenominator
	txtByNum = .ByNumerator
	txtByDen = .ByDenominator
20	If .IsIndependent Then
	chkIsIndependent = 1
	Else
ra ra	chkIsIndependent = 0
	End If
295	If .MixedNumbers Then
	chkMixedNumbers = 1
⊞ ₽≅n;	Else
4. 	chkMixedNumbers = 0
78.7 27.1	End If
30	End With
	mudtType = vtFraction
	· -
	Case "VarString"
	Set mudtVarString = mudtVar
35	With mudtVarString
	mudtType = vtString
	If .Delimiter = Chr(STRING_DELIMITER) The
	' do nothing
	Else
40	ConvertDelimiter
	.Delimiter = Chr(STRING_DELIMITER)
	End If
	'load list view control
	If .IsIndexed Then
45	chkIndexed = 1

```
Else
                      chkIndexed = 0
                   End If
                    LoadListView
 5
                 End With
               Case "VarUntyped"
                 Set mudtVarUntyped = mudtVar
                 mudtType = vtUntyped
10
            End Select
            mudtOldType = mudtType
            cboVarType.ListIndex = mudtType 'generates a cboVarType_Click event
15
          Else ' it's an add
            mudtType = vtInteger
            mudtOldType = mudtType
            cboVarType.ListIndex = vtInteger 'generates a cboVarType_Click event
End If
          'changes control states if model is frozen
          UpdateControlStates
 End Sub
 Private Sub cmdVarOK Click()
  is i
 ' will capitalize the first letter of the variable name, if it's not
          'capitalized already.
30
          txtVariableName LostFocus
          ' make sure all input is valid, otherwise, make 'em fix it!
          If ValidateForm = False Then
            Exit Sub
          End If
35
          If mbytAddEditFlag = aeEdit Then 'we're editing an old one
            Call ProcessEdit
          Else
            Call ProcessAdd
40
          End If
```

```
Unload Me
       End Sub
       Private Sub cmdVarCancel Click()
 5
          Unload Me
       End Sub
       Private Sub cmdVarImport_Click()
10
          Dim strFN As String
          With cdlCD
            .FileName = ""
            .DialogTitle = "Import strings from file"
15
            .Filter = "String Files (*.str)|*.str|"
            .DefaultExt = ".str"
 ű,
            .InitDir = "c:\tcs\tca\strings"
 đ١
            .Flags = cdlOFNFileMustExist + cdlOFNHideReadOnly
 IJ
            On Error GoTo Cancel
2Q:
            .ShowOpen
            On Error GoTo 0
            strFN = .FileName
          End With
          On Error GoTo BeatIt ' trap open, I/O errors
          Open strFN For Input Access Read As 1
30
          Dim varR As Variant
          Dim varIndexed As Variant
          Dim varNumIndices As Variant
          Dim strMessage As String
          Dim mcolStr As Collection
35
          Dim intI As Integer
          Input #1, varIndexed
          If varIndexed Then
40
            strMessage = "indexed."
          Else
            strMessage = "not indexed."
```

```
End If
          If varIndexed ⇔ chkIndexed Then
             Call MsgBox("Unable to import: file contains string values that are " & _
 5
               strMessage, vbExclamation, "Error")
             GoTo BeatIt
          End If
          Input #1, varNumIndices
10
          Do
             Input #1, varR
             If varIndexed Then
               Set mcolStr = New Collection
15
               Call mcolStr.Add(varR)
               For intI = 1 To varNumIndices - 1
                  Input #1, varR
                  Call mcolStr.Add(varR)
               Next intI
               Call AddColToListView(mcolStr)
             Else
 Men of the Cast Cast
               Call AddStrToListView(varR)
             End If
25
          Loop Until EOF(1)
 4D
        BeatIt:
          Close 1
        Cancel:
          Exit Sub
        End Sub
        Private Sub cmdVarExport_Click()
35
          Dim strFN As String
          cdlCD.CancelError = True
40
          With cdlCD
             .FileName = ""
             .DialogTitle = "Export strings to file"
             .Filter = "String Files (*.str)|*.str|"
             .DefaultExt = ".txt"
```

```
.InitDir = "c:\tcs\tca\strings"
            .Flags = cdlOFNOverwritePrompt + cdlOFNHideReadOnly
            On Error GoTo Cancel
            .ShowSave
 5
            On Error GoTo 0
            strFN = .FileName
          End With
          On Error GoTo BeatIt
10
          Open strFN For Output Access Write As 1
          Dim varW As Variant
          varW = chkIndexed 'so we can tell if it's indexed
15
          Print #1, varW
          varW = lvwStrings.ColumnHeaders.Count ' how many indices
          Print #1, varW
20
          Dim intl As Integer
          Dim intI2 As Integer
          Dim lsiItem As ListItem
          intI = 1
          Do' write the data
            Set lsiItem = lvwStrings.ListItems.Item(intI)
            varW = lsiItem.Text
 ij,
            Print #1, varW
30
            If chkIndexed Then
               For intI2 = 1 To lvwStrings.ColumnHeaders.Count - 1
                 varW = lsiItem.SubItems(intI2)
                 Print #1, varW
               Next intI2
35
            End If
            intI = intI + 1
40
          Loop Until intI > lvwStrings.ListItems.Count
        BeatIt:
          Close 1
        Cancel:
```

```
End Sub
       Private Sub IvwStrings MouseDown(Button As Integer, Shift As Integer, _
 5
          X As Single, Y As Single)
          If Button = vbRightButton Then
            PopupMenu mnuString
          End If
       End Sub
10
       Private Sub mnuStringAdd_Click()
          If chkIndexed Then
            With frmIndexedString
               ' set the model
               .Model = mudtModel
               ' set the edit flag
               .AddEditFlag = aeAdd
               ' set var name
               .VariableName = txtVariableName
               ' do it
               .Show vbModal
               If .OK Then
                 Call AddColToListView(.SubStringCollection)
               End If
            End With
          Else
            With frmString
               ' set the model
               .Model = mudtModel
               ' set the string
30
               .StringValue = ""
               ' set var name
               .VariableName = txtVariableName
               ' do it
               .Show vbModal
35
               If .OK Then
                 Call AddStrToListView(.StringValue)
               End If
            End With
```

Exit Sub

40

End If

UpdateControlStates

End Sub

Private Sub mnuStringEdit_Click()

5 Dim colC As Collection

If lvwStrings.SelectedItem Is Nothing Then Exit Sub ' Make sure list item is selected

```
If chkIndexed Then
10
             With frmIndexedString
               ' set the model
               .Model = mudtModel
               ' set the edit flag
               .AddEditFlag = aeEdit
               ' set the substring collection
15
               .SubStringCollection = GetSubStringCollection(lvwStrings.SelectedItem)
               ' set var name
 .VariableName = txtVariableName
 ű
               ' do it
 <u>a</u>
201
               .Show vbModal
 Mer allen Mer allen
               If .OK Then
                  Call UpdateListView(lvwStrings.SelectedItem, .SubStringCollection)
               End If
             End With
          Else
             With frmString
               ' set the model
               .Model = mudtModel
               ' set the string
               .StringValue = lvwStrings.SelectedItem
               ' set var name
               .VariableName = txtVariableName
               ' do it
               .Show vbModal
35
               If .OK Then
                  Set colC = New Collection
                  Call colC.Add(.StringValue)
                  Call UpdateListView(lvwStrings.SelectedItem, colC)
               End If
40
             End With
          End If
```

End Sub

	Private Sub mnuStringRemove_Click()
	If IvwStrings.SelectedItem Is Nothing Then Exit Sub
5	If MsgBox("Remove string value " & lvwStrings.SelectedItem.Text & "?", _ vbQuestion + vbYesNo) = vbNo Then Exit Sub End If
	With lvwStrings Call .ListItems.Remove(.SelectedItem.index) End With
10	UpdateControlStates
	End Sub
	Private Sub chkIsIndependent_Click()
15	Call FormatForm
a from Say	End Sub
	Private Sub cboVarType_Click()
He House	mudtType = cboVarType.ListIndex
201	Call FormatForm
	End Sub
5 1	Private Sub txtVariableName_GotFocus()
25	' Automatically select all text when TextBox gets focus Call txtSelectAll(txtVariableName)
	End Sub
30	Private Sub txtVariableName_LostFocus()
	Dim strName As String Dim udtVar As Variable
	'Capitalize the variable name in the textbox strName = txtVariableName

Call CapitalizeString(strName) txtVariableName = strName
End Sub
Private Sub txtFrom_GotFocus()

Pri

' Automatically select all text when TextBox gets focus Call txtSelectAll(txtFrom)

End Sub

5

Private Sub txtTo GotFocus() 10

> ' Automatically select all text when TextBox gets focus Call txtSelectAll(txtTo)

End Sub

[] 1**5**] Private Sub txtBy_GotFocus() Men alber Men Alon Man Can Call

' Automatically select all text when TextBox gets focus Call txtSelectAll(txtBy)

End Sub

Private Sub txtFromNum_GotFocus()

' Automatically select all text when TextBox gets focus Call txtSelectAll(txtFromNum)

End Sub

20 4) 4)

25 Private Sub txtFromDen GotFocus()

> ' Automatically select all text when TextBox gets focus Call txtSelectAll(txtFromDen)

End Sub

30 Private Sub txtToNum GotFocus()

> ' Automatically select all text when TextBox gets focus Call txtSelectAll(txtToNum)

> > VBSCA -240-

	Private Sub txtToDen_GotFocus()
5	' Automatically select all text when TextBox gets focus Call txtSelectAll(txtToDen)
	End Sub
	Private Sub txtByNum_GotFocus()
	'Automatically select all text when TextBox gets focus Call txtSelectAll(txtByNum)
10	End Sub
	Private Sub txtByDen_GotFocus()
Mr.	'Automatically select all text when TextBox gets focus Call txtSelectAll(txtByDen)
	End Sub
1	Private Sub FormatForm()
a	cmdVarImport.Visible = False cmdVarExport.Visible = False
	chkIsIndependent.TabStop = False
ļai ma	txtFrom.TabStop = False txtTo.TabStop = False
	txtBy.TabStop = False
25	txtFromNum.TabStop = False
30	txtFromDen.TabStop = False
	txtToNum.TabStop = False
	txtToDen.TabStop = False
	txtByNum.TabStop = False
	txtByDen.TabStop = False
	lvwStrings.TabStop = False chkTrailingZeros.TabStop = False
	chkTrailingZeros.TabStop = False chkTrailingZeros.TabStop = False
	chkMixedNumbers.TabStop = False
	omination things of the state o

End Sub

35

Select Case mudtType

	Case villlegel
	fraFractionRange.Visible = False
	fraFractionFormat.Visible = False
	fraIndependent.ZOrder
5	fraIntRealRange.ZOrder
	fraRealFormat.Visible = False
	chkIsIndependent.TabStop = True
	If chkIsIndependent Then
	fraIntRealRange.Visible = True
10	txtFrom.TabStop = True
	txtTo.TabStop = True
	txtBy.TabStop = True
	Else
	fraIntRealRange.Visible = Fals
15	End If
	Case vtReal
	fraFractionRange.Visible = False
(T 8	fraFractionFormat.Visible = False
20	fraIndependent.ZOrder
01	fraIntRealRange.ZOrder
LII	fraRealFormat.ZOrder
20 III III III III III III III III III I	fraRealFormat.Visible = True
Ú)	chkIsIndependent.TabStop = True
25	If chkIsIndependent Then
	fraIntRealRange.Visible = True
5 7	txtFrom.TabStop = True
L . 54	txtTo.TabStop = True
4.) 71	txtBy.TabStop = True
30.	Else
	fraIntRealRange.Visible = Fals
- 2 3	End If
	chkOnGrid.TabStop = True
	chkTrailingZeros.TabStop = True
35	
	Case vtFraction
	fraIntRealRange.Visible = False
	fraRealFormat.Visible = False
	fraIndependent.ZOrder
40	fraFractionRange.ZOrder
	fraFractionFormat.ZOrder
	fraFractionFormat.Visible = True
	chkIsIndependent.TabStop = True
	If chkIsIndependent Then
45	fraFractionRange Visible = Tru

	txtFromNum.TabStop = True txtFromDen.TabStop = True
	txtToNum.TabStop = True
_	txtToDen.TabStop = True
5	txtByNum.TabStop = True
	txtByDen.TabStop = True
	Else
	fraFractionRange.Visible = False
10	End If
10	chkMixedNumbers.TabStop = True
	Case vtString
	fraString.ZOrder
	cmdVarImport.Visible = True
15	cmdVarExport.Visible = True
·	Case vtUntyped
	fraUntyped.ZOrder
	P 101 /
2 U)	End Select
UI III	Dim intTabIndex As Integer
20 C C C C C C C C C C C C C C C C C C C	Dim incraomdex As integer
T.	intTabIndex = 4
25	In Tuonidox
W	Call AddTab(chkIsIndependent, intTabIndex)
9 	Call AddTab(txtFrom, intTabIndex)
9 C	Call AddTab(txtTo, intTabIndex)
4. #4	Call AddTab(txtBy, intTabIndex)
30	Call AddTab(txtFromNum, intTabIndex)
C)	Call AddTab(txtFromDen, intTabIndex)
	Call AddTab(txtToNum, intTabIndex)
	Call AddTab(txtToDen, intTabIndex)
	Call AddTab(txtByNum, intTabIndex)
35	Call AddTab(txtByDen, intTabIndex)
	Call AddTab(chkTrailingZeros, intTabIndex)
	Call AddTab(chkOnGrid, intTabIndex)
	Call AddTab(chkMixedNumbers, intTabIndex)
	End Sub
40	'add a tab, if its active
	Private Sub AddTab(ByVal ctlC As Control, intIndex As Integer)
	If ctlC.TabStop Then

```
ctlC.TabIndex = intIndex
            intIndex = intIndex + 1
          End If
       End Sub
 5
       Private Function ValidateForm() As Boolean
          ValidateForm = False
          'check variable name length > 0
          If Len(txtVariableName) = 0 Then
            Call MsgBox("Variable names must be 1 or more characters long.", _
10
               vbExclamation, "Error")
            txtVariableName.SetFocus
            Exit Function
          End If
15
          'check first character for alpha
          If Asc(txtVariableName) < 65 Or Asc(txtVariableName) > 91 Then
 ij,
            Call MsgBox("Variable names must begin in a letter", _
 Πî
              vbExclamation, "Error")
 U1
            txtVariableName.SetFocus
20.
            Exit Function
          End If
          ' check for unique variable name
          Dim blnUnique As Boolean
          blnUnique = True
          Select Case mbytAddEditFlag
            Case aeAdd
               blnUnique = mudtModel.Variables.UniqueName(txtVariableName)
30
            Case aeEdit
               blnUnique = mudtModel.Variables.UniqueName(txtVariableName, 1, mudtVar)
          End Select
          If blnUnique = False Then
            Call MsgBox("Variable name is already in use.", vbExclamation, "Error")
            txtVariableName.SetFocus
35
            Exit Function
          End If
```

```
' if integer or real, validate contents of From, To, By
          If cboVarType = "Integer" Or cboVarType = "Real" Then
            If Not ValidateRange Then
               Call MsgBox("Entries in From, To, and By must be either a number " &
 5
                 "or a string variable containing a numeric value. " & _
                 "Expressions or math variables are not permitted.", _
                 vbExclamation, "Error")
               Exit Function
10
            End If
          End If
          ValidateForm = True
        End Function
       Private Function ValidateRange() As Boolean
15
          Dim conC As Control
          Dim colC As New Collection
 41
          Dim udtV As Variable
 <u>a</u>
          Dim udtVS As VarString
 Ul
20
          Dim intl As Integer
          Dim blnOK As Boolean
          Call colC.Add(txtFrom)
          Call colC.Add(txtTo)
          Call colC.Add(txtBy)
          For Each conC In colC
            blnOK = False
            If IsNumeric(conC) Then
               blnOK = True
30
            Else ' see if the box contains a string variable
               For Each udtV In mudtModel.Variables
                 If udtV.Typ = vtString Then
                   Set udtVS = udtV
35
                   If udtVS.IsIndexed Then
                      For intI = 1 To udtVS.NumIndices
                        If conC = GetIndexedName(udtV.name, intl) Then
                          blnOK = True
                          Exit For
40
                        End If
                      Next intl
                   ElseIf conC = udtV.name Then
```

	blnOK = True
	End If
	End If
	If blnOK Then
5	Exit For
	End If
	Next udtV
	End If
	If Not blnOK Then
10	ValidateRange = False
	Exit Function
	End If
	Next conC
15	ValidateRange = True
	· · · · · · · · · · · · · · · · · · ·
	End Function
	Private Sub ProcessEdit()
mą	v
tas k¶i	'Check to see if the type has changed
2	If mudtType
Ü	
m ga	With mlstListBox
45	' remove the old variable from the collection
## ## ## ## ## ## ## ## ## ## ## ## ##	Call mudtModel.Variables.Remove(Str(.ItemData(.ListIndex)))
25	' add the new variable
e e	Call AddVariable
L .)	' update the index in the list box
##.	.ItemData(.ListIndex) = mudtVar.index
₩# Ka	' replace the text in the list box
30 լ	.List(.ListIndex) = mudtVar.ScreenFormat
	End With
	Else
	'update it with new data from form
35	Select Case mudtType
	Case vtInteger
	Call mudtVarInt.Update(txtVariableName, _
40	txtFrom, txtTo, txtBy, _
40	chkIsIndependent, chkChecksum)
	Care wtDeel
	Case vtReal
	Call mudtVarReal.Update(txtVariableName,
	txtFrom, txtTo, txtBy, chkIsIndependent, _

chkChecksum, chkTrailingZeros.Value, cboPrecision, chkOnGrid)

```
Case vtFraction
                 Call mudtVarFraction.Update(txtVariableName, _
                   txtFromNum, txtFromDen, txtToNum, txtToDen,
 5
                   txtByNum, txtByDen, chkIsIndependent, chkChecksum, _
                   chkMixedNumbers)
              Case vtString
                 Dim intl As Integer
10
                 Dim intI2 As Integer
                 Dim colStr As Collection
                 Dim udtSS As SubString
15
                 mudtVar.name = txtVariableName
                 mudtVar.Checksum = chkChecksum
                 mudtVarString.IsIndexed = chkIndexed
                 'build a new collection of strings
                 Set colStr = New Collection
203
                 With lywStrings
                   For intI = 1 To (.ListItems.Count)
                      Set udtSS = New SubString
                      udtSS.Delimiter = mudtVarString.Delimiter
                      Call udtSS.AddSubString(.ListItems.Item(intI).Text)
                      For intI2 = 1 To .ColumnHeaders.Count - 1
                        Call udtSS.AddSubString(.ListItems.Item(intI).SubItems(intI2))
                     Next intI2
 ű
                      Call colStr.Add(udtSS.StringValue)
                   Next intI
3Q£
                 End With
 C)
                 mudtVarString.StringCollection = colStr
            End Select
35
            With mlstListBox
              ' replace the text in the list box
              .List(.ListIndex) = mudtVar.ScreenFormat
            End With
40
          End If
       End Sub
```

45

Private Sub ProcessAdd()

VBSCA -247-

With mlstListBox ' Add the new variable to the variable list box Call .AddItem(mudtVar.ScreenFormat) 'Set ItemData to index value of the variable object 5 .ItemData(.ListCount - 1) = mudtVar.index ' Check the check box .Selected(.ListCount - 1) = True End With 10 End Sub Private Sub AddVariable() ' Add the new variable Select Case mudtType 15 Case vtInteger Set mudtVar = mudtModel.Variables.AddInteger(txtVariableName, _ True, txtFrom, txtTo, txtBy, chkIsIndependent, 10 U1 chkChecksum) 20 Case vtReal Set mudtVar = mudtModel.Variables.AddReal(txtVariableName, True, txtFrom, txtTo, txtBy, chkIsIndependent, _ chkChecksum, chkTrailingZeros.Value, cboPrecision, chkOnGrid) Case vtFraction Set mudtVar = mudtModel.Variables.AddFraction(txtVariableName, _ True, txtFromNum, txtFromDen, txtToNum, txtToDen, txtByNum, txtByDen, chkIsIndependent, chkChecksum, _ chkMixedNumbers) 30 Case vtString Dim intI As Integer Dim intI2 As Integer Dim colStr As New Collection 35 Dim udtSS As SubString With lvwStrings For intI = 1 To (.ListItems.Count) Set udtSS = New SubString 40 udtSS.Delimiter = Chr(STRING DELIMITER) udtSS.AddSubString (.ListItems.Item(intI).Text)

Call AddVariable

```
For intI2 = 1 To .ColumnHeaders.Count - 1
                     Call udtSS.AddSubString(.ListItems.Item(intI).SubItems(intI2))
                   Next intI2
                 Call colStr.Add(udtSS.StringValue)
 5
                 Next intI
              End With
              Set mudtVar = mudtModel.Variables.AddString(txtVariableName, True,
              chkChecksum, Chr(STRING DELIMITER), chkIndexed, colStr)
10
            Case vtUntyped
              Set mudtVar = mudtModel.Variables.AddUntyped(txtVariableName, True,
                 chkChecksum)
          End Select
15
        End Sub
        Private Sub UpdateControlStates()
          Dim conC As Control
 Ō1
          On Error Resume Next
          ' shut off all controls that have an enabled property
          For Each conC In Me
            If mudtModel.IsFrozen Then
              conC.Enabled = False
            Else
              conC.Enabled = True
            End If
          Next conC
          On Error GoTo 0
          ' these stay on even if model is frozen
30
          cmdVarCancel.Enabled = True
          fraString.Enabled = True
          lvwStrings.Enabled = True
          cmdEdit.Enabled = True
          mnuStringEdit.Enabled = True
35
          'if model is frozen, change caption of edit button, menu to browse
          If mudtModel.IsFrozen Then
            cmdEdit.Caption = "Browse"
            mnuStringEdit.Caption = "Browse"
```

```
End If
          ' turn export on if there's something to export
          cmdVarExport.Enabled = CBool(lvwStrings.ListItems.Count)
          'shut off "edit", "remove" buttons, menus if the listview is empty
 5
          If lvwStrings.ListItems.Count = 0 Then
            mnuStringEdit.Enabled = False
            cmdEdit.Enabled = False
            mnuStringRemove.Enabled = False
10
            cmdRemove.Enabled = False
          End If
        End Sub
        'this is used to convert version 0.6 indexed strings to version 0.7 style
       Private Sub ConvertDelimiter()
15
          Dim colStr As Collection
          Dim varS As Variant
          With mudtVarString
            Set colStr = .StringCollection
            For Each varS In colStr
               varS = ReplaceAll(varS, .Delimiter, Chr(STRING_DELIMITER))
            Next varS
          End With
       End Sub
       Private Sub LoadListView()
          Dim intl As Integer
          Dim varS As Variant
30
          With mudtVarString
            If chkIndexed Then
               build column headers
               For intI = 1 To .NumIndices - 1
                 Call lvwStrings.ColumnHeaders.Add(, , _
35
                   Str(intI), lvwStrings.Width / 4)
               Next intI
            End If
            ' fill in values
```

```
For Each varS In .StringCollection
               Call AddStrToListView(varS)
            Next varS
          End With
 5
       End Sub
       Private Sub AddColToListView(ByVal colS As Collection)
          Dim lsiLI As ListItem
          Set lsiLI = lvwStrings.ListItems.Add(, , "")
          Call UpdateListView(lsiLI, colS)
10
        End Sub
       Private Sub AddStrToListView(ByVal strS As String)
          Dim udtSS As New SubString
15
          Dim IsiLI As ListItem
          Dim intI As Integer
 Mr. Am
          Set lsiLI = lvwStrings.ListItems.Add(, , "")
          udtSS.Delimiter = Chr(STRING DELIMITER)
          udtSS.StringValue = strS
          Call UpdateListView(lsiLI, udtSS.StringCollection)
       End Sub
 <u>ļ</u>_L
        Private Sub UpdateListView(ByVal lsiLI As ListItem, ByVal colS As Collection)
25
          Dim intI As Integer
          Dim intW As Integer
          Dim strColHeading As String
          If chkIndexed Then
30
            intW = 4
          Else
            intW = 1
          End If
          ' expand the number of columns if there aren't enough
35
          For intI = lvwStrings.ColumnHeaders.Count To colS.Count - 1
            If chkIndexed Then
               strColHeading = Str(intI + 1)
```

```
Call lvwStrings.ColumnHeaders.Add(,, strColHeading,_
                  lvwStrings.Width / intW)
             Else
               strColHeading = " "
 5
                Call lvwStrings.ColumnHeaders.Add(, , strColHeading)
             End If
           Next intI
           ' plug in the values
10
           For intI = 1 To colS.Count
             If intI = 1 Then
               lsiLI = colS.Item(intI)
             Else
               lsiLI.SubItems(intI - 1) = colS.Item(intI)
15
             End If
          Next intI
           ' get rid of anything in the list view past colS. Count
          For intI = colS.Count + 1 To lvwStrings.ColumnHeaders.Count
20
             If intI > 1 Then
               lsiLI.SubItems(intI - 1) = ""
             Else
               lsiLI = ""
             End If
          Next intI
          Dim blnEmpty As Boolean
          ' get rid of columns with all "" from right to left
          ' stop when first column with any string > 0 length is encountered
          For intI = lvwStrings.ColumnHeaders.Count To 1 Step -1
             For Each IsiLI In lvwStrings.ListItems
               blnEmpty = True
               If intI > 1 Then
35
                  If lsiLI.SubItems(intI - 1) <> "" Then
                    blnEmpty = False
                    Exit For
                  End If
               ElseIf lsiLI <> "" Then
40
                 blnEmpty = False
                  Exit For
               End If
             Next lsiLI
             If blnEmpty Then
45
               Call lvwStrings.ColumnHeaders.Remove(intI)
```

```
Else
               Exit For
            End If
          Next intI
 5
          Dim intI2 As Integer
          ' get rid of rows with "" in all columns from the bottom up
          For intI2 = lvwStrings.ListItems.Count To 1 Step -1
            Set lsiLI = lvwStrings.ListItems.Item(intI2)
10
            For intI = 1 To lvwStrings.ColumnHeaders.Count
               blnEmpty = True
               If intI > 1 Then
                 If lsiLI.SubItems(intI - 1) 	<> "" Then
15
                   blnEmpty = False
                   Exit For
                 End If
               ElseIf lsiLI <> "" Then
                 blnEmpty = False
                 Exit For
20
               End If
            Next intI
            If blnEmpty Then
               Call lvwStrings.ListItems.Remove(intI2)
            End If
          Next intI2
 End Sub
        Private Function GetSubStringCollection(ByVal lsiLI As ListItem) As Collection
 ļa i
303
          Dim colC As New Collection
          Dim intl As Integer
          Call colC.Add(lsiLI)
35
          For intI = 1 To lvwStrings.ColumnHeaders.Count - 1
            Call colC.Add(lsiLI.SubItems(intI))
          Next intI
          Set GetSubStringCollection = colC
40
        End Function
```

```
' Application.cls
       VERSION 1.0 CLASS
       BEGIN
        MultiUse = -1 'True
 5
        Persistable = 0 'NotPersistable
        DataBindingBehavior = 0 'vbNone
        DataSourceBehavior = 0 'vbNone
        MTSTransactionMode = 0 'NotAnMTSObject
       END
       Attribute VB_Name = "TCAApplication"
10
       Attribute VB GlobalNameSpace = False
       Attribute VB Creatable = True
       Attribute VB PredeclaredId = False
       Attribute VB Exposed = False
       Attribute VB_Ext_KEY = "SavedWithClassBuilder", "Yes"
15
       Attribute VB Ext KEY = "Top Level", "Yes"
       Option Explicit
 đì
       Public Sub Run()
         Dim udtP As New Prolog
         If udtP.StartProlog("hlp4lib.p4") = False Then
            Call MsgBox("Prolog failure on startup", vbExclamation, "Error")
          End If
         frmTCA.Show
       End Sub
```

	'CClones.cls VERSION 1.0 CLASS BEGIN
5	MultiUse = -1 'True END
3	Attribute VB_Name = "CClones" Attribute VB_GlobalNameSpace = False Attribute VB_Creatable = True Attribute VB_PredeclaredId = False
10	Attribute VB_Exposed = False Option Explicit
	' enable i/o Private mudtFile As File
15	'to hold collection Private mcolClones As Collection
	' the sequence number appended to clone filenames Private mintSeqNum As Integer
office British office of the control	' is dirty Private mblnIsDirty As Boolean
20 ¹⁷	Private Sub Class_Initialize()
	'creates the collection when this class is created Set mcolClones = New Collection
	End Sub
25	Private Sub Class_Terminate()
	'destroys collection when this class is terminated Set mcolClones = Nothing
	End Sub
30	Public Property Get Item(vntIndexKey As Variant) As Clone
	'used when referencing an element in the collection 'vntIndexKey contains either the Index or Key to the collection, 'this is why it is declared as a Variant 'Syntax: Set foo = x.Item(xyz) or Set foo = x.Item(5)

```
Set Item = mcolClones(vntIndexKey)
        End Property
        Public Property Get Count() As Long
          'used when retrieving the number of elements in the
 5
          'collection. Syntax: Debug.Print x.Count
          Count = mcolClones.Count
        End Property
10
        Public Property Get NextSeqNum() As Integer
          mintSeqNum = mintSeqNum + 1
          NextSeqNum = mintSeqNum
          mblnIsDirty = True
15
        End Property
 ·I]
20°
        Public Property Let SeqNum(ByVal intNewValue As Integer)
          mintSeqNum = intNewValue
          mblnIsDirty = True
 L'A M. L'A
        End Property
        Public Property Get SeqNum() As Integer
 ļ.
          SeqNum = mintSeqNum
25
        End Property
       Public Property Get IsDirty() As Boolean
          Dim udtClone As Clone
30
          ' see if any collection members are dirty
          If Not mblnIsDirty Then
            For Each udtClone In mcolClones
              If udtClone.IsDirty Then
                 mblnIsDirty = True
                 Exit For
35
```

```
End If
            Next udtClone
          End If
          IsDirty = mblnIsDirty
 5
       End Property
       Private Function NextID() As Long
          ' creates a unique index to associate a clone and a listbox
          Static IngID As Long
10
          lngID = lngID + 1
          NextID = lngID
15
       End Function
       Public Function Add(ByVal strFN As String,
          Optional ByVal blnAddSeqNum = False) As Clone
 Ų,
          Dim udtClone As New Clone
          ' add the clone sequence number to the file name if blnAddSeqNum is True.
          If blnAddSeqNum Then
            udtClone.FileName = left(strFN, Len(strFN) - 4) & _
              Trim(Str(NextSeqNum)) & ".doc"
          Else
            udtClone.FileName = ExtractFileName(strFN)
          End If
 C1
          udtClone.Index = NextID
30
          ' use index of the clone as the key
          Call mcolClones.Add(udtClone, Str(udtClone.Index))
          Set Add = udtClone
       End Function
       Public Function AddObj(ByVal udtClone As Clone) As Clone
35
          udtClone.Index = NextID
          ' use index of the clone as the key
```

Call mcolClones.Add(udtClone, Str(udtClone.Index)) Set AddObj = udtClone **End Function** Public Sub Remove(vntIndexKey As Variant) 5 'used when removing an element from the collection 'vntIndexKey contains either the Index or Key, which is why 'it is declared as a Variant 'Syntax: x.Remove(xyz) mcolClones.Remove vntIndexKey 10 mblnIsDirty = True End Sub Public Property Get NewEnum() As IUnknown Attribute NewEnum.VB_UserMemId = -4 'this property allows you to enumerate 'this collection with the For...Each syntax Set NewEnum = mcolClones.[NewEnum] **End Property** 17.3 ff., 17.1 1.2 ff., 17.1 Public Sub Clear() 'empties the collection class į. C) Set mcolClones = Nothing Set mcolClones = New Collection 25 mblnIsDirty = True End Sub Public Sub ReadCollection(ByVal strFN As String, ByVal lngStartIndex As Long, _ 30 ByVal lngEndIndex As Long) Set mudtFile = New File mudtFile.FileName = strFN Call mudtFile.ReadFile(Me, lngStartIndex, lngEndIndex) 35 VBSCA -258-

```
Set mudtFile = Nothing
       End Sub
 5
       Public Sub ReadObjects()
          Dim udtClone As Clone
          On Error GoTo BeatIt
10
          Do Until Err. Number <> 0
            Set udtClone = New Clone
            Call udtClone.ReadObjectData(mudtFile)
            udtClone.Index = NextID
            Call mcolClones.Add(udtClone, Str(udtClone.Index))
15
          Loop
       BeatIt:
20
          Exit Sub
       End Sub
       Public Function WriteCollection(ByVal strFN As String,
          ByVal lngIndexPos As Long, ByVal lngSeekPos) As Long
 ij,
          Set mudtFile = New File
25=
 mudtFile.FileName = strFN
          WriteCollection = mudtFile.WriteFile(Me, False, lngIndexPos, lngSeekPos)
          Set mudtFile = Nothing
30
          mblnIsDirty = False
       End Function
       Public Sub WriteObjects()
35
          Dim udtClone As Clone
          For Each udtClone In mcolClones
```

Call udtClone.WriteObjectData(mudtFile)
Next udtClone

End Sub

```
'CConstraints.cls
        VERSION 1.0 CLASS
        BEGIN
         MultiUse = -1 'True
 5
        END
        Attribute VB Name = "CConstraints"
        Attribute VB_GlobalNameSpace = False
        Attribute VB Creatable = True
        Attribute VB PredeclaredId = False
        Attribute VB Exposed = False
10
        Option Explicit
        ' enable i/o
        Private mudtFile As New File
        'local variable to hold collection
       Private mcolConstraint As Collection
15
        ' is dirty
 ď,
 Private mblnIsDirty As Boolean
 Ben offen Han offen
        Public Property Let IsDirty(ByVal blnNewValue As Boolean)
          mblnIsDirty = blnNewValue
        End Property
       Public Property Get IsDirty() As Boolean
          Dim udtCon As Constraint
 For Each udtCon In mcolConstraint
            If udtCon.IsDirty Then
25
               mblnIsDirty = True
               Exit For
            End If
          Next udtCon
30
          IsDirty = mblnIsDirty
       End Property
       Private Sub Class Initialize()
```

	'creates the collection when this class is created Set mcolConstraint = New Collection
	End Sub
5	Private Sub Class_Terminate()
	'destroys collection when this class is terminated Set mcolConstraint = Nothing
	End Sub
10	Public Property Get Item(vntIndexKey As Variant) As Constraint
1 5	'used when referencing an element in the collection 'vntIndexKey contains either the Index or Key to the collection, 'this is why it is declared as a Variant 'Syntax: Set foo = x.Item(xyz) or Set foo = x.Item(5) Set Item = mcolConstraint(vntIndexKey)
The state of the s	End Property
. 25	Public Property Get Count() As Long
15000000000000000000000000000000000000	'used when retrieving the number of elements in the 'collection. Syntax: Debug.Print x.Count Count = mcolConstraint.Count
	End Property
	Public Sub AddObject(udtCon As Constraint)
25 25	' adds constraint objects directly to the collection
	udtCon.Index = NextID Call mcolConstraint.Add(udtCon, Str(udtCon.Index))
20	mblnIsDirty = True
30	End Sub
	Public Function Add(ByVal strConstraint As String, ByVal blnEnabled As Boolean, _ ByVal udtType As ConstraintType, ByVal strComment As String) As Constraint
35	'create a new object

	Dim objNewMember As Constraint Set objNewMember = New Constraint
	'set the properties passed into the method
	With objNewMember
5	.ConstraintString = strConstraint
	.Enabled = blnEnabled
	.ConstraintType = udtType
	.Comment = strComment
	.Index = NextID
10	' add the new object to the collection
	Call mcolConstraint.Add(objNewMember, Str\$(.Index))
	End With
	'return the object created
15	Set Add = objNewMember
	Set objNewMember = Nothing
E.	mblnIsDirty = True
w] Mi	End Function
Ui Ui	End I unction
20 mm m m m m m m m m m m m m m m m m m	Public Sub Remove(vntIndexKey As Variant)
ijļ	
d e E¶	'used when removing an element from the collection
*	'vntIndexKey contains either the Index or Key, which is why
<u>C</u> j	'it is declared as a Variant
<u>.</u>	'Syntax: x.Remove(xyz)
23	mcolConstraint.Remove vntIndexKey
jeh Få	mblnIsDirty = True
<u>1</u>	momistrity True
, marie	End Sub
30	Public Function NewEnum() As IUnknown
	Attribute NewEnum.VB_UserMemId = -4
	Attribute NewEnum.VB_MemberFlags = "40"
	'this property allows you to enumerate
	'this collection with the ForEach syntax
35	Set NewEnum = mcolConstraint.[_NewEnum]
	End Function
	End Punction
	Private Function NextID() As Long

```
'creates a unique index to associate a constraint and the constraint listbox(es)
          Static IngID As Long
          lngID = lngID + 1
 5
          NextID = lngID
        End Function
        'returns true if strCon is already a constraint in the collection. Used
        ' when importing constraints to make sure dups are not introduced.
        Public Function UniqueConstraint(ByVal strCon As String) As Boolean
10
          Dim udtCon As Constraint
          UniqueConstraint = True
          'Check for duplicate constraint
15
          For Each udtCon In mcolConstraint
            If strCon = udtCon.ConstraintString Then
               UniqueConstraint = False
 Ø١
               Exit For
20 7 7 7 7
            End If
          Next udtCon
        End Function
25<u>1</u>
        Public Sub ReadCollection(ByVal strFN As String, ByVal lngStartIndex As Long, _
          ByVal lngEndIndex As Long)
          mudtFile.FileName = strFN
          Call mudtFile.ReadFile(Me, lngStartIndex, lngEndIndex)
        End Sub
        Public Sub ReadObjects()
30
          Dim udtCon As Constraint
          On Error GoTo BeatIt
35
          Do Until Err. Number <> 0
            Set udtCon = New Constraint
            Call udtCon.ReadObjectData(mudtFile)
            udtCon.Index = NextID
```

Loop BeatIt: 5 Exit Sub **End Sub** Public Function WriteCollection(ByVal strFN As String, ByVal lngIndexPos As Long, ByVal lngSeekPos) As Long 10 mudtFile.FileName = strFN WriteCollection = mudtFile.WriteFile(Me, False, lngIndexPos, lngSeekPos) mblnIsDirty = False 15 **End Function** Public Sub WriteObjects() Dim udtCon As Constraint For Each udtCon In mcolConstraint Call udtCon.WriteObjectData(mudtFile) Next udtCon 25] End Sub Public Sub Clear(ByVal udtType As VariableType) 'empties the collection class of all constraints of type udtType Dim udtCon As Constraint For Each udtCon In mcolConstraint 30 If udtCon.ConstraintType = udtType Then Call mcolConstraint.Remove(Str(udtCon.Index)) End If 35 Next udtCon

Call mcolConstraint.Add(udtCon, Str(udtCon.Index))

End Sub

```
'returns true if an enabled string variable name was used
        ' in any enabled constraint
        Public Function StringVarNamesUsed(ByVal udtCVar As CVariables) As Boolean
 5
          'First create a collection of all enabled constraint strings
          Dim udtCon As Constraint
          Dim colStrings As New Collection
10
          For Each udtCon In mcolConstraint
            If udtCon.Enabled Then
               colStrings.Add udtCon.ConstraintString
            End If
          Next udtCon
15
          ' create a variable collection with variable names sorted in length
          ' from longest to shortest
 41
          Dim udtSCVar As CVariables
          Set udtSCVar = udtCVar.SortVarNamesByLength
          'nibble variable names out of the string collection, using enabled
          'variable names sorted in length from longest to shortest
          Dim vntS As Variant
          Dim vntT As Variant
          Dim vntStart As Variant
          Dim udtVar As Variable
 ļ
          For Each vntS In colStrings
301
            For Each udtVar In udtSCVar
               If udtVar.Enabled Then
                 vntStart = InStr(1, vntS, udtVar.Name)
                 If vntStart Then
                    If udtVar.Typ = vtString Then
                      StringVarNamesUsed = True
35
                      Exit Function
                    Else
                      vntT = vntS
                      vntS = left(vntT, vntStart - 1) & _
                        right(vntT, Len(vntT) - vntStart - __
40
                        Len(udtVar.Name) + 1)
                    End If
                 End If
```

End If Next udtVar Next vntS

5 StringVarNamesUsed = False

End Function

```
'Checksum.cls
        VERSION 1.0 CLASS
        BEGIN
         MultiUse = -1 'True
 5
        END
        Attribute VB Name = "Checksum"
        Attribute VB GlobalNameSpace = False
        Attribute VB Creatable = True
        Attribute VB PredeclaredId = False
        Attribute VB Exposed = False
10
        Option Explicit
       Private mcolStr As Collection
       Private Sub Class_Initialize()
          Set mcolStr = New Collection
15
       End Sub
 Mary aller Mary allers Harm Mary
       Public Sub AddValue(ByVal strNewValue As String)
          Call mcolStr.Add(strNewValue)
       End Sub
       Public Function ComputeCS() As Double
20.
          Dim n As Integer
          Dim dblCS As Double
          Dim dblSum As Double
          Dim varStr As Variant
          Dim cntr As Integer
          Dim dblT As Double
25
          cntr = 1
          On Error GoTo Overflow
30
          For Each varStr In mcolStr
            dblSum = 0
            n = Len(varStr)
            While n > 0
               dblSum = Asc(Mid(varStr, n, 1)) * n + dblSum
35
```

Wish rolling there, reflects them thereby places It will be the transfer than the course of the transfer than the transf

The state of the s

	' Clone.cls VERSION 1.0 CLASS
5	BEGIN MultiUse = -1 'True END
10	Attribute VB_Name = "Clone" Attribute VB_GlobalNameSpace = False Attribute VB_Creatable = True Attribute VB_PredeclaredId = False Attribute VB_Exposed = False Option Explicit
	' current version of data produced by this class Const mintVERSIONSTAMP As Integer = 1
15	' file name (without path) of this clone Private mstrFN As String
en (fin) (fin. 1)" en and han th	' hold document handle Private mdocCloneDoc As Document
անա (Մարավիա ն « ո առա ո ո ո	' checksum of variables Private mdblChecksum As Double
	' index Private mlngIndex As Long
	' is dirty Private mblnIsDirty As Boolean
25	' has been routed to TCS Private mbytIsRouted As Byte
	' program Private mudtProgram As Program
	' domain Private mudtDomain As Domain
30	' the batch id Private mstrBatchID As String
	' the target template Private udtDeliveryMode As DeliveryMode

	' pure or real model Private mudtNature As Nature
	'TDer's estimate of difficulty (1-5) Private mbytTDEstimate As Byte
5	' difficulty has been calculated Private mbytIsDifficultyCalculated As Byte
	' the key Private mstrKey As String
10	' the item type Private mudtItemType As ItemType
	Public Enum Domain doArithmetic = 0 doAlgebra = 1 doDataAnalysis = 2 doGeometry = 3 End Enum
15, 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	Public Enum Nature naPure = 0 naReal = 1 End Enum
18.11 18.18 (18.2 18.18 18.18 18.18 18.18 18.18 18.18 18.18 18.18 18.18 18.18 18.18 18.18 18.18 18.18 18.1	' difficulty estimate Private mudtDE As DifficultyEstimate
	Private Sub Class_Initialize()
	mblnIsDirty = False
23	End Sub
	Public Property Get FileName() As String
	FileName = mstrFN
30	End Property
	Public Property Let FileName(ByVal strNewValue As String)
	If mstrFN ⇔ strNewValue Then

```
mstrFN = strNewValue
           mblnIsDirty = True
         End If
       End Property
       Public Property Get CloneDoc() As Document
 5
         Set CloneDoc = mdocCloneDoc
       End Property
       Public Property Let CloneDoc(ByVal docNewValue As Document)
10
         Set mdocCloneDoc = docNewValue
       End Property
       Public Property Get Checksum() As Double
 41
         Checksum = mdblChecksum
 Ø1
15
       End Property
       Public Property Let Checksum(ByVal dblNewValue As Double)
         If mdblChecksum <> dblNewValue Then
           mdblChecksum = dblNewValue
           mblnIsDirty = True
         End If
       End Property
       Public Property Get Index() As Long
         Index = mlngIndex
       End Property
25
       Public Property Let Index(ByVal lngNewValue As Long)
         If mlngIndex ⇔ lngNewValue Then
           mlngIndex = lngNewValue
           mblnIsDirty = True
         End If
```

```
End Property
       Public Property Get IsDirty() As Boolean
         IsDirty = False
 5
         If IsDifficultyCalculated Then 'don't check DE if difficultly hasn't been calculated!
           If mblnIsDirty Or mudtDE.IsDirty Then
              IsDirty = True
           End If
10
         Else
           If mblnIsDirty Then
              IsDirty = True
           End If
         End If
15
       End Property
       Public Property Get IsRouted() As Byte
 Ľ.
 Ō١
         IsRouted = mbytIsRouted
 Ľ٦
20
       End Property
       Public Property Let IsRouted(ByVal bytNewValue As Byte)
         If mbytIsRouted 	<> bytNewValue Then
           mbytIsRouted = bytNewValue
           mblnIsDirty = True
         End If
       End Property
       Public Property Get Program() As Program
         Program = mudtProgram
       End Property
30
       Public Property Let Program(ByVal udtNewValue As Program)
         mudtProgram = udtNewValue
           mblnIsDirty = True
```

```
End If
       End Property
      Public Property Get Domain() As Domain
         Domain = mudtDomain
 5
      End Property
      Public Property Let Domain(ByVal udtNewValue As Domain)
         mudtDomain = udtNewValue
           mblnIsDirty = True
         End If
10
      End Property
 C,
      Public Property Get IsDifficultyCalculated() As Byte
 4)
 G1
         IsDifficultyCalculated = mbytIsDifficultyCalculated
 Ų1
15]
      End Property
 ų)
      Public Property Let IsDifficultyCalculated(ByVal bytNewValue As Byte)
        mbytIsDifficultyCalculated = bytNewValue
           mblnIsDirty = True
        End If
      End Property
      Public Property Get TDEstimate() As Byte
         TDEstimate = mbytTDEstimate
      End Property
25
      Public Property Let TDEstimate(ByVal bytNewValue As Byte)
         If mbytTDEstimate >> bytNewValue Then
           mbytTDEstimate = bytNewValue
          mblnIsDirty = True
```

```
End If
       End Property
       Public Property Get BatchID() As String
         BatchID = mstrBatchID
 5
       End Property
       Public Property Let BatchID(ByVal strNewValue As String)
         If mstrBatchID <> strNewValue Then
            mstrBatchID = strNewValue
            mblnIsDirty = True
10
         End If
       End Property
 C)
       Public Property Get Key() As String
 111
 ۵ì
15
         Key = mstrKey
       End Property
       Public Property Let Key(ByVal strNewValue As String)
         If mstrKey 			 strNewValue Then
            mstrKey = strNewValue
            mblnIsDirty = True
         End If
       End Property
       Public Property Get ItemType() As ItemType
         ItemType = mudtItemType
       End Property
       Public Property Let ItemType(ByVal udtNewValue As ItemType)
         If mudtItemType \simeq udtNewValue Then
25
            mudtItemType = udtNewValue
            mblnIsDirty = True
```

```
End If
                            End Property
                            Public Property Get DeliveryMode() As DeliveryMode
                                    DeliveryMode = udtDeliveryMode
    5
                            End Property
                             Public Property Let DeliveryMode(ByVal udtNewValue As DeliveryMode)
                                    udtDeliveryMode = udtNewValue
                                             mblnIsDirty = True
10
                                     End If
                            End Property
    Public Property Get Nature() As Nature
    4Ï
    Q1
13 mg and a district of the second of the se
                                    Nature = mudtNature
                             End Property
                            Public Property Let Nature(ByVal udtNewValue As Nature)
                                    If mudtNature <> udtNewValue Then
                                             mudtNature = udtNewValue
                                             mblnIsDirty = True
                                     End If
   End Property
                            Public Property Get DiffEst() As DifficultyEstimate
                                     Set DiffEst = mudtDE
                             End Property
                             Public Property Let DiffEst(ByVal udtNewValue As DifficultyEstimate)
                                     Set mudtDE = udtNewValue
25
                                     mblnIsDirty = True
```

	End Property
	Public Sub OpenDoc(ByVal udtWord As MSWord, ByVal strPath As String)
	Dim udtDS As New DocStatus
5	If udtDS.IsOpen(mstrFN) = False Then Set mdocCloneDoc = _ udtWord.WordApp.Documents.Open(FileName:=strPath & mstrFN) End If
10	mdocCloneDoc.Activate
10	End Sub
	Public Sub CloseDoc()
	Dim udtDS As New DocStatus
երը այդ այդ այի այդ	If udtDS.IsOpen(mstrFN) Then Call mdocCloneDoc.Close(wdSaveChanges) ' save changes Set mdocCloneDoc = Nothing End If
	End Sub
	Public Sub ReadObjectData(udtFile As File)
T.	Dim vField As Variant
20 m m m m m m	Call udtFile.ReadField(vField) ' returns the version stamp Call udtFile.ReadField(vField) FileName = ExtractFileName(vField)
25	Call udtFile.ReadField(vField) Key = ExtractFileName(vField) Call udtFile.ReadField(vField) ItemType = ExtractFileName(vField)
30	Call udtFile.ReadField(vField) Program = vField Call udtFile.ReadField(vField) Domain = vField Call udtFile.ReadField(vField)
35	BatchID = vField Call udtFile.ReadField(vField)

	DeliveryMode = vField Call addFile ReadField(vField)
	Call udtFile.ReadField(vField)
	Nature = vField
_	Call udtFile.ReadField(vField)
5	TDEstimate = vField
	Call udtFile.ReadField(vField)
	IsRouted = vField
	Call udtFile.ReadField(vField)
	IsDifficultyCalculated = vField
10	Set mudtDE = Nothing
	If IsDifficultyCalculated Then
	Select Case Program
	Case prGRE
	Set mudtDE = New GREDifficultyEstimate
15	Case prGMAT
	Set mudtDE = New GMATDifficultyEstimate
	End Select
	Call mudtDE.ReadObjectData(udtFile)
₽#?	End If
20,	
n.	End Sub
119	
72 22 a	Public Sub WriteObjectData(udtFile As File)
	• • • • • • • • • • • • • • • • • • • •
20 mm mm ng mg ng	Call udtFile.WriteField(mintVERSIONSTAMP)
25	Call udtFile.WriteField(ExtractFileName(mstrFN))
2	Call udtFile.WriteField(Key)
	Call udtFile.WriteField(ItemType)
	Call udtFile.WriteField(Program)
Ę]	Call udtFile.WriteField(Domain)
# [] **** [] ***** [] ******************	Call udtFile.WriteField(BatchID)
	Call udtFile.WriteField(DeliveryMode)
ئ ية	Call udtFile.WriteField(Nature)
	Call udtFile.WriteField(TDEstimate)
	Call udtFile.WriteField(IsRouted)
35	Call udtFile.WriteField(IsDifficultyCalculated)
55	If IsDifficultyCalculated Then
	Call mudtDE.WriteObjectData(udtFile)
	End If
	Elid II
40	mblnIsDirty = False
. •	
	End Sub

	' CModels.cls
	VERSION 1.0 CLASS
	BEGIN
	MultiUse = -1 'True
5	END
	Attribute VB_Name = "CModels"
	Attribute VB_GlobalNameSpace = False
	Attribute VB_Creatable = True
10	Attribute VB_PredeclaredId = False
10	Attribute VB_Exposed = False
	Option Explicit
	'to hold collection
	Private mcolModels As Collection
	Private Sub Class_Initialize()
15	'creates the collection when this class is created
43	Set mcolModels = New Collection
15.1 "	End Sub
	Life 540
¥3	Private Sub Class Terminate()
eģu Va	_
2∯ື່	'destroys collection when this class is terminated
≅ f=1	Set mcolModels = Nothing
144 144 111	T 101
	End Sub
ļe la	Duklia Duanauty Cat Itam/ymtIndayVay As Variant) As Madal
	Public Property Get Item(vntIndexKey As Variant) As Model
25 25	'used when referencing an element in the collection
23	'vntIndexKey contains either the Index or Key to the collection,
	'this is why it is declared as a Variant
	'Syntax: Set foo = $x.Item(xyz)$ or Set foo = $x.Item(5)$
	Set Item = mcolModels(vntIndexKey)
30	, , , , , , , , , , , , , , , , , , ,
	End Property
	Dublic Duamouty Cat Count() As Long
	Public Property Get Count() As Long
	'used when retrieving the number of elements in the
	'collection. Syntax: Debug.Print x.Count
35	Count = mcolModels.Count

	Public Sub AddObject(udtMod As Model)
5	'adds model objects directly to the collection. Use the file name as the 'key.
	Call mcolModels.Add(udtMod, Str(udtMod.FileName))
	End Sub
10	Public Function AddNew(ByVal strFN As String, _ ByVal udtItemType As ItemType) As Model
	Dim udtMod As Model Dim udtSMC As SMCModel Dim udtQC As QCModel
15	Dim udtDS As DSModel
15. T.	Select Case udtItemType
ráu ru fri	Case ptStandardMC
197	Set udtSMC = New SMCModel
20: - 11	Set udtMod = udtSMC
	Case ptQuantComp
<u>C</u>	Set $udtQC = New QCModel$
	Set udtMod = udtQC
21 THE TOTAL	Case ptDataSuff
<u>L.</u> j	Set udtDS = New DSModel
ta d	Set udtMod = udtDS
30	End Select
50	' file name has full path
	udtMod.FileName = strFN
	udtMod.IsFrozen = False
35	' strip path from key
	Call mcolModels.Add(udtMod, ExtractFileName(strFN))

Set AddNew = udtMod

End Property

End Function

Public Function AddExisting(ByVal strFN As String, _
ByVal udtItemType As ItemType) As Model

Dim udtMod As New Model

Dim udtSMC As SMCModel

Dim udtQC As QCModel

Dim udtDS As DSModel

Select Case udtItemType

10 Case ptStandardMC
Set udtSMC = New SMCModel
Set udtMod = udtSMC

Case ptQuantComp Set udtQC = New QCModel Set udtMod = udtQC

Case ptDataSuff
Set udtDS = New DSModel
Set udtMod = udtDS

End Select
' file name has full path
udtMod.FileName = strFN
Call udtMod.ReadModel

' strip path from key
Call mcolModels.Add(udtMod, ExtractFileName(strFN))

Set AddExisting = udtMod

30 End Function

15

¥Ĵ

O U

20:

35

Public Sub Remove(vntIndexKey As Variant)

'used when removing an element from the collection
'vntIndexKey contains either the Index or Key, which is why
'it is declared as a Variant
'Syntax: x.Remove(xyz)
mcolModels.Remove vntIndexKey

End Sub

Public Property Get NewEnum() As IUnknown Attribute NewEnum.VB_UserMemId = -4 Attribute NewEnum.VB_MemberFlags = "40"

Public Sub Clear()

10 'empties the collection class

Set mcolModels = Nothing Set mcolModels = New Collection

End Sub

Hen alben then albe of the tent the first 1978 starts

```
'Constraint.cls
        VERSION 1.0 CLASS
        BEGIN
         MultiUse = 0 'False
 5
         Persistable = 0 'NotPersistable
         DataBindingBehavior = 0 'vbNone
         DataSourceBehavior = 0 'vbNone
         MTSTransactionMode = 0 'NotAnMTSObject
        END
        Attribute VB Name = "Constraint"
10
        Attribute VB GlobalNameSpace = False
        Attribute VB Creatable = True
        Attribute VB PredeclaredId = False
        Attribute VB Exposed = False
        Attribute VB_Ext_KEY = "SavedWithClassBuilder", "Yes"
15
        Attribute VB Ext KEY = "Member0", "CloningConstraint"
        Attribute VB Ext KEY = "Member1", "DifficultyConstraint"
        Attribute VB_Ext_KEY = "Member2", "MathConstraint"
Attribute VB_Ext_KEY = "Member3", "VariableDefinition"
 C)
        Attribute VB Ext KEY = "Top Level", "Yes"
201
 Ļή
        Option Explicit
 Men dien Men den
        'current version of data produced by this class
        Const mintVERSIONSTAMP As Integer = 1
       Private mudtType As VariableType
        Private mstrConstraint As String
        Private mstrComment As String
        Private mlngIndex As Long
 <u>L</u>
       Private mblnEnabled As Boolean
 C)
       Private mblnIsDirty As Boolean
 30
       'These numbers correspond to the indices of the constraint listboxes in frmTCA
        Public Enum ConstraintType
          ctVariation = 0
          ctDistractor = 1
        End Enum
       Public Property Get ConstraintString() As String
35
          ConstraintString = mstrConstraint
        End Property
```

```
Public Property Let ConstraintString(ByVal strNewValue As String)
          If mstrConstraint <> strNewValue Then
            mstrConstraint = strNewValue
            mblnIsDirty = True
          End If
 5
       End Property
       Public Property Get Comment() As String
          Comment = mstrComment
10
       End Property
       Public Property Let Comment(ByVal strNewValue As String)
          If mstrComment <> strNewValue Then
            mstrComment = strNewValue
15
            mblnIsDirty = True
          End If
 Men alben Men alben Men Men Men
       End Property
       Public Property Get ConstraintType() As ConstraintType
20
1
          ConstraintType = mudtType
       End Property
       Public Property Let ConstraintType(ByVal udtNewValue As ConstraintType)
          mudtType = udtNewValue
25
            mblnIsDirty = True
          End If
       End Property
30
       Public Property Get index() As Long
          index = mlngIndex
       End Property
```

Public Property Let index(ByVal lngNewValue As Long) mlngIndex = lngNewValue **End Property** 5 Public Property Get Enabled() As Boolean Enabled = mblnEnabled **End Property** Public Property Let Enabled(ByVal blnNewValue As Boolean) 10 If mblnEnabled \Leftrightarrow blnNewValue Then mblnEnabled = blnNewValue mblnIsDirty = True End If **End Property** We offer for the first for the Public Property Let IsDirty(ByVal blnNewValue As Boolean) mblnIsDirty = blnNewValue **End Property** Public Property Get IsDirty() As Boolean IsDirty = mblnIsDirty **End Property** Public Sub Update(ByVal strConstraint As String, ByVal udtType As ConstraintType, ByVal strComment As String) 25 ConstraintString = strConstraint ConstraintType = udtType Comment = strComment 30 End Sub Public Sub ReadObjectData(udtFile As File) Dim vField As Variant

	Call udtFile.ReadField(vField) ' read version stamp Call udtFile.ReadField(vField) ConstraintType = vField
5	Call udtFile.ReadField(vField) Enabled = vField
10	Call udtFile.ReadField(vField) ConstraintString = vField
	Call udtFile.ReadField(vField) Comment = vField
	End Sub
15	Public Sub WriteObjectData(udtFile As File)
1. The Control of the	Call udtFile.WriteField(mintVERSIONSTAMP) Call udtFile.WriteField(ConstraintType) Call udtFile.WriteField(Enabled) Call udtFile.WriteField(ConstraintString) Call udtFile.WriteField(Comment)
20 10 10 10 10 10 10 10 10 10 10 10 10 10	mblnIsDirty = False
	End Sub
25	' makes a copy of this object Public Function Copy() As Constraint
	Dim udtC As New Constraint
	udtC.Enabled = Enabled udtC.index = index
30	<pre>udtC.IsDirty = IsDirty udtC.ConstraintType = ConstraintType udtC.ConstraintString = ConstraintString udtC.Comment = Comment</pre>
35	Set Copy = udtC

End Function

VBSCA -287-

	'ConstraintSolver.cls
	VERSION 1.0 CLASS
	BEGIN
	MultiUse = -1 'True
5	Persistable = 0 'NotPersistable
	DataBindingBehavior = 0 'vbNone
	DataSourceBehavior = 0 'vbNone
	MTSTransactionMode = 0 'NotAnMTSObject
	END
10	Attribute VB_Name = "ConstraintSolver"
10	-
	Attribute VB_GlobalNameSpace = False
	Attribute VB_Creatable = True
	Attribute VB_PredeclaredId = False
	Attribute VB_Exposed = False
15	Option Explicit
	Private mcolVs As Collection
= 1	Private mcolVsSave As Collection
	Private mcolCs As Collection
17979	Private mcolCsSave As Collection
201	Private mcolValues As Collection
∯a Ia	Private mbytDiffWeight As Byte
4)	Private mdblChecksum As Double
	Private mintIndex As Integer
20 mm and	•
8	Private WithEvents mwudtP As Prolog
25	Attribute mwudtP.VB VarHelpID = -1
44	Private mlngRet As Long
jes jes	Private mblnPrologIsRunning As Boolean
25.25	
lans!	Public Enum SolveRequester
	srTest = 0
30	srGenerate = 1
50	End Enum
	Elia Elialii
	Public Enum SolveReturn
	srNoSolutions = 0
	srSuccess = 1
25	
35	srPrologAborted = -1
	srPrologError = -2
	End Enum

Private mudtSolveRequester As SolveRequester

	Private Sub Class_Initialize()
5	Set mcolVs = New Collection Set mcolVsSave = New Collection Set mcolCs = New Collection Set mcolCsSave = New Collection Set mcolValues = New Collection
	End Sub
	Private Sub Class_Terminate()
10	'Kill Prolog Set mwudtP = Nothing
	End Sub
	Public Property Let Prolog(ByVal udtNewValue As Prolog)
	Set mwudtP = udtNewValue
131 U1	End Property
and the second s	Public Property Let DiffWeight(ByVal bytNewValue As Byte)
	mbytDiffWeight = bytNewValue
	End Property
201	Public Sub AddVariable(ByVal udtNewValue As Variable)
	If udtNewValue.Enabled Then Call mcolVs.Add(udtNewValue.Copy) ' uses a copy of the variable Call mcolVsSave.Add(udtNewValue.Copy) End If
25	End Sub
	Public Sub AddConstraint(ByVal udtNewValue As Constraint)
•	If udtNewValue.Enabled Then Call mcolCs.Add(udtNewValue.Copy) ' uses a copy of the constraint
30	Call mcolCsSave.Add(udtNewValue.Copy) '

Public Function GetNextValue(strVarName As String, _ strValue As String) As Boolean Dim udtVal As Value 5 If mintIndex <= mcolValues.Count Then Set udtVal = mcolValues.Item(mintIndex) strVarName = udtVal.VariableName strValue = udtVal.Value ' if the value is ^, replace with ^^ so Word doesn't choke 10 If strValue = "^" Then strValue = "^^" mintIndex = mintIndex + 1GetNextValue = True Else GetNextValue = False 15 End If End Function Public Sub ResetValueIndex() mintIndex = 120^[] End Sub Public Property Get Checksum() Checksum = mdblChecksum **End Property** 25 Public Function Solve(ByVal udtSolveRequester As SolveRequester) As SolveReturn Dim udtVal As Value Dim udtC As Constraint Dim udtV As Variable Dim udtVS As VarString Dim udtSS As StringSolver 30 mudtSolveRequester = udtSolveRequester Set mcolValues = New Collection 35 mintIndex = 1

End Sub

If mcolValues.Count = 0 Then Solve = srNoSolutions5 **Exit Function** End If 'solve all string variables For Each udtV In mcolVs 10 If udtV.Typ = vtString Then Set udtVS = udtV' if this variable has no strings, error If udtVS.StringCollection.Count = 0 Then 15 Solve = srNoSolutions**Exit Function** End If Set udtSS = New StringSolver udtSS.StringVariable = udtVS Call LoadStringValues(udtVS, udtSS) End If Next udtV ij 'resolve any nested values for all string variable names 25 ResolveNestedStrings 41 'resolve string variable names embedded in math variable ranges ResolveStringsInMathVariables ' resolve string variable names embedded in constraints ResolveConstraints ' set the difference weight (difference between variants) mwudtP.DiffWeight = mbytDiffWeight 35 Dim blnMathToSolve As Boolean ' add non-string variables to prolog via the value object collection For Each udtVal In mcolValues If Not udtVal.VariableType = vtString Then 40 Call mwudtP.AddVariable(udtVal.PrologString) blnMathToSolve = True End If Next udtVal

CreateValueCollection

45

```
' add all constraints
          For Each udtC In mcolCs
            Call mwudtP.AddConstraint(udtC.ConstraintString)
            blnMathToSolve = True
 5
          Next udtC
          'call prolog if there are math constraints, error if no solution found
          If blnMathToSolve Then
             ' get rid of the kill file if it exists
10
            DestroyKillFile
            mblnPrologIsRunning = True
            'runs async, notifies this class when it's done via the Finished event
            mwudtP.SolveConstraintsRandomly
            If udtSolveRequester = srTest Then
               frmProlog.Caption = "Testing constraints"
15
               frmProlog.lblProlog.Caption = "Click Abort to terminate this test."
               frmProlog.Show vbModal
            Else
               Do
20
                 DoEvents
               Loop While mblnPrologIsRunning
            End If
            If frmProlog.Abort Then
               ' create the kill file
               CreateKillFile
               Solve = srPrologAborted
               Exit Function
            End If
            'not aborted
            Select Case mlngRet
               Case Is < 0
                 Solve = srPrologError
                 Call MsgBox("Prolog error: " & Str(mlngRet), vbExclamation, "Error")
                 Exit Function
               Case 0
35
                 Solve = srNoSolutions
                 Exit Function
            End Select
          End If
40
          ' load up values from Prolog
          For Each udtVal In mcolValues
            If Not udtVal.VariableType = vtString Then
               udtVal.Value = mwudtP.Value(udtVal.VariableName)
45
            End If
```

Next udtVal ' resolve string values that are math variable names ResolveMathVariablesInStrings Dim udtChecksum As New Checksum ' compute the checksum of values For Each udtVal In mcolValues If udtVal.Checksum Then Call udtChecksum.AddValue(udtVal.Value) End If Next udtVal mdblChecksum = udtChecksum.ComputeCS Solve = srSuccess'restore the variable and constraint collections their original states, ' as substitutions may have contaminated them. Set mcolVs = New Collection Set mcolCs = New Collection For Each udtV In mcolVsSave Call mcolVs.Add(udtV.Copy) Next udtV For Each udtC In mcolCsSave Call mcolCs.Add(udtC.Copy) Next udtC **End Function** 'this event raised in Prolog class Private Sub mwudtP Finished(ByVal lngRet As Long) mblnPrologIsRunning = False mlngRet = lngRet

End Sub

End If

'kill the form if this is a test

frmProlog.Kill

If mudtSolveRequester = srTest Then

5

10

1**5**

30

35

Private Sub CreateValueCollection() Dim intI As Integer Dim udtV As Variable Dim udtVS As VarString 5 Dim udtVal As Value For Each udtV In mcolVs If udtV.Typ = vtString Then Set udtVS = udtV10 If udtVS.IsIndexed Then For intI = udtVS.NumIndices To 1 Step -1 Set udtVal = New ValueudtVal.VariableName = GetIndexedName(udtV.name, intI) udtVal.VariableType = udtV.Typ 15 udtVal.Checksum = udtV.Checksum udtVal.PrologString = udtV.PrologFormat Call mcolValues.Add(udtVal, udtVal.VariableName) Next intI Else J. Set udtVal = New Value20m U1 udtVal.VariableName = udtV.name 25 udtVal.VariableType = udtV.Typ udtVal.Checksum = udtV.Checksum udtVal.PrologString = udtV.PrologFormat Call mcolValues.Add(udtVal, udtVal.VariableName) End If Else Set udtVal = New Value udtVal.VariableName = udtV.name udtVal.VariableType = udtV.Typ 3**0** j udtVal.Checksum = udtV.Checksum udtVal.PrologString = udtV.PrologFormat Call mcolValues.Add(udtVal, udtVal.VariableName) End If 35 Next udtV End Sub Private Sub LoadStringValues(ByVal udtV As Variable, ByVal udtSS As StringSolver) 40 Dim intl As Integer Dim varS As Variant

Dim strVN As String

```
Dim udtVal As Value
          Dim udtVS As VarString
          Set udtVS = udtV
 5
          ' get the value or values (if indexed)
          If udtVS.IsIndexed Then
            intI = 1
            For Each varS In udtSS.RandomValueCollection
              strVN = GetIndexedName(udtV.name, intI)
10
              Set udtVal = mcolValues.Item(strVN)
              udtVal.Value = varS
              intI = intI + 1
            Next varS
15
          Else
            Set udtVal = mcolValues.Item(udtV.name)
            udtVal.Value = udtSS.RandomValueCollection(1)
          End If
2Qj
        End Sub
 O1
       Private Sub ResolveNestedStrings()
          Dim blnContinue As Boolean
          Dim udtVal As Value
          Do
            blnContinue = False
            For Each udtVal In mcolValues
              If udtVal.VariableType = vtString Then
                 If ResolveString(udtVal.VariableName) Then
                   blnContinue = True
30₃
                 End If
              End If
            Next udtVal
          Loop Until blnContinue = False
35
        End Sub
       Private Function ResolveString(ByVal strVN As String) As Boolean
          Dim udtVal As Value
          Dim udtVal2 As Value
40
          Dim strT As String
```

```
ResolveString = False
         For Each udtVal In mcolValues
            If udtVal.VariableType = vtString Then
              Set udtVal2 = mcolValues.Item(strVN)
 5
              strT = ReplaceAll(udtVal.Value, strVN, udtVal2.Value)
              udtVal.Value = strT
                ResolveString = True
10
              End If
            End If
         Next udtVal
       End Function
15
       Private Sub ResolveStringsInMathVariables()
         Dim udtVal As Value
         Dim udtVal2 As Value
 ű
         For Each udtVal In mcolValues
 ۵ì
201
            If udtVal.VariableType = vtString Then
              For Each udtVal2 In mcolValues
 Mr. An Mr.
                If Not udtVal2.VariableType = vtString Then
                  udtVal2.PrologString = ReplaceAll(udtVal2.PrologString,
                     udtVal.VariableName, udtVal.Value)
                End If
              Next udtVal2
            End If
         Next udtVal
       End Sub
3₫
       Private Sub ResolveConstraints()
         Dim udtC As Constraint
         Dim udtVal As Value
35
         For Each udtVal In mcolValues
            If udtVal.VariableType = vtString Then
              For Each udtC In mcolCs
                udtC.ConstraintString = ReplaceAll(udtC.ConstraintString,
                   udtVal.VariableName, udtVal.Value)
40
              Next udtC
            End If
```

```
End Sub
       Private Sub ResolveMathVariablesInStrings()
          Dim udtVal As Value
          Dim udtVal2 As Value
 5
          For Each udtVal In mcolValues
            If udtVal. VariableType = vtString Then
              For Each udtVal2 In mcolValues
                 If Not udtVal2. VariableType = vtString Then
10
                   udtVal.Value = ReplaceAll(udtVal.Value, udtVal2.VariableName,
                     udtVal2.Value)
                 End If
              Next udtVal2
            End If
15
          Next udtVal
 ď)
        End Sub
 Ø١
 Ų1
20
       'CVariables.cls
        VERSION 1.0 CLASS
       BEGIN
        MultiUse = -1 'True
       END
       Attribute VB Name = "CVariables"
       Attribute VB GlobalNameSpace = False
       Attribute VB Creatable = True
        Attribute VB PredeclaredId = False
        Attribute VB Exposed = False
       Attribute VB Ext KEY = "SavedWithClassBuilder", "Yes"
       Attribute VB_Ext_KEY = "Collection", "Variable"
30
        Attribute VB Ext KEY = "Member0", "Variable"
       Attribute VB Ext KEY = "Top Level", "Yes"
       Option Explicit
       'enable i/o
35
       Private mudtFile As File
       'to hold collection
```

Private mcolVariable As Collection

Next udtVal

	' is dirty Private mblnIsDirty As Boolean
	Public Property Let IsDirty(ByVal blnNewValue As Boolean)
	mblnIsDirty = blnNewValue
5	End Property
	Public Property Get IsDirty() As Boolean
	Dim udtVar As Variable
10	For Each udtVar In mcolVariable If udtVar.IsDirty Then mblnIsDirty = True Exit For End If Next udtVar
15 15	IsDirty = mblnIsDirty
15:	End Property
700 100 100 100 100 100 100 100 100 100	Private Sub Class_Initialize()
Ħ	'creates the collection when this class is created Set mcolVariable = New Collection
	Set mudtFile = New File
	End Sub
25	Private Sub Class_Terminate()
	'destroys collection when this class is terminated Set mcolVariable = Nothing
30	'destroys the File object Set mudtFile = Nothing
	End Sub
	Public Property Get Item(vntIndexKey As Variant) As Variable

5	'vntIndexKey contains either the Index or Key to the collection, 'this is why it is declared as a Variant 'Syntax: Set foo = x.Item(xyz) or Set foo = x.Item(5) Set Item = mcolVariable(vntIndexKey)
	End Property
	Public Property Get Count() As Long
10	'used when retrieving the number of elements in the 'collection. Syntax: Debug.Print x.Count Count = mcolVariable.Count
	End Property
15	Public Sub AddObject(udtVar As Variable)
	' adds variable objects directly to the collection
440 H 442	udtVar.Index = NextID Call mcolVariable.Add(udtVar, Str(udtVar.Index))
20	End Sub
	Public Function AddInteger(ByVal strName As String, ByVal blnEnabled As Boolean, _ ByVal strFrom As String, ByVal strTo As String, ByVal strBy As String, _ ByVal blnIsIndependent As Boolean, ByVal blnChecksum As Boolean) As Variable
E) þi	'create a new object
2 <u>5</u> j	Dim udtVar As Variable Dim udtVarInteger As New VarInteger
	Set udtVar = udtVarInteger
30	'set the properties passed into the method With udtVar .Typ = vtInteger
	.Name = strName .Enabled = blnEnabled .Index = NextID
35	.Checksum = blnChecksum End With
	With udtVarInteger

```
.From = strFrom
            .Too = strTo
            .By = strBy
            .IsIndependent = blnIsIndependent
 5
          End With
          ' add the new object to the collection
          Call mcolVariable.Add(udtVarInteger, Str(udtVar.Index))
          'return the object created
          Set AddInteger = udtVarInteger
10
        End Function
        Public Function AddReal(ByVal strName As String, ByVal blnEnabled As Boolean,
          ByVal strFrom As String, ByVal strTo As String, ByVal strBy As String, _
          ByVal blnIsIndependent As Boolean, ByVal blnChecksum As Boolean,
          ByVal blnTrailingZeros As Boolean,
15
          ByVal strPrecision As String, ByVal blnOnGrid As Boolean) As Variable
 ďĵ
          'create a new object
          Dim udtVar As Variable
 Ų٦
20==
          Dim udtVarReal As New VarReal
 ij,
          Set udtVar = udtVarReal
          'set the properties passed into the method
          With udtVar
            .Typ = vtReal
            .Name = strName
            .Enabled = blnEnabled
            .Index = NextID
            .Checksum = blnChecksum
30
          End With
          With udtVarReal
            .From = strFrom
            .Too = strTo
35
            .By = strBy
            .IsIndependent = blnIsIndependent
            .TrailingZeros = blnTrailingZeros
            .Precision = strPrecision
            .IsOnGrid = blnOnGrid
```

End With

40

' add the new object to the collection Call mcolVariable.Add(udtVarReal, Str(udtVar.Index)) 'return the object created Set AddReal = udtVarReal **End Function** Public Function AddFraction(ByVal strName As String, ByVal blnEnabled As Boolean, ByVal strFromNum As String, ByVal strFromDen As String, ByVal strToNum As String, ByVal strToDen As String, ByVal strByNum As String, ByVal strByDen As String, ByVal blnIsIndependent As Boolean, ByVal blnChecksum As Boolean, ByVal blnMixedNumbers As Boolean) As Variable 'create a new object Dim udtVar As Variable Dim udtVarFraction As New VarFraction Set udtVar = udtVarFraction'set the properties passed into the method With udtVar 2**6**] .Typ = vtFraction.Name = strName .Enabled = blnEnabled .Index = NextID.Checksum = blnChecksum End With With udtVarFraction .FromNumerator = strFromNum .FromDenominator = strFromDen .ToNumerator = strToNum .ToDenominator = strToDen .ByNumerator = strByNum .ByDenominator = strByDen .IsIndependent = blnIsIndependent .MixedNumbers = blnMixedNumbers End With ' add the new object to the collection Call mcolVariable.Add(udtVarFraction, Str(udtVar.Index))

5

10

15

30

35

40

'return the object created

Set AddFraction = udtVarFraction

End Function Public Function AddString(ByVal strName As String, ByVal blnEnabled As Boolean, ByVal blnChecksum As Boolean, ByVal strDelimiter As String, ByVal blnIsIndexed As Boolean, ByVal colString As Collection) As Variable 5 'create a new object Dim udtVar As Variable Dim udtVarString As New VarString 10 Set udtVar = udtVarString'set the properties passed into the method With udtVar .Typ = vtString15 .Name = strName .Enabled = blnEnabled .Index = NextID۵ì .Checksum = blnChecksum U1 End With 20 udtVarString.Delimiter = strDelimiter udtVarString.StringCollection = colString udtVarString.IsIndexed = blnIsIndexed ' add the new object to the collection Call mcolVariable.Add(udtVarString, Str(udtVar.Index)) 'return the object created Set AddString = udtVarString **End Function** 30 Public Function AddUntyped(ByVal strName As String, ByVal blnEnabled As Boolean, ByVal blnChecksum As Boolean) 'create a new object Dim udtVar As Variable 35 Dim udtVarUntyped As New VarUntyped Set udtVar = udtVarUntyped

'set the properties passed into the method

```
With udtVar
            .Typ = vtUntyped
            .Name = strName
            .Enabled = blnEnabled
 5
            .Index = NextID
            .Checksum = blnChecksum
          End With
          ' add the new object to the collection
          Call mcolVariable.Add(udtVarUntyped, Str(udtVar.Index))
10
          'return the object created
          Set AddUntyped = udtVarUntyped
        End Function
        Public Sub Remove(vntIndexKey As Variant)
          'used when removing an element from the collection
20年5月
          'vntIndexKey contains either the Index or Key, which is why
          'it is declared as a Variant
          'Syntax: x.Remove(xyz)
          mcolVariable.Remove vntIndexKey
          mblnIsDirty = True
        End Sub
 Public Property Get NewEnum() As IUnknown
        Attribute NewEnum.VB UserMemId = -4
25
        Attribute NewEnum.VB MemberFlags = "40"
          'this property allows you to enumerate
          'this collection with the For...Each syntax
          Set NewEnum = mcolVariable.[ NewEnum]
30
       End Property
       Private Function NextID() As Long
          ' creates a unique index to associate a variable and the variable listbox
          Static IngID As Long
35
          lngID = lngID + 1
```

```
End Function
       'returns true if strName is already a variable name in the collection. If the
 5
        'optional parameter is used, the function will not check that variable for a dup.
        Public Function UniqueName(ByVal strName As String, _
          Optional ByVal bytSkipThisVar As Byte = 0,
          Optional ByVal udtSkipVar As Variable) As Boolean
          Dim udtVar As Variable
10
          UniqueName = True
          'Check for duplicate variable name
          For Each udtVar In mcolVariable
15
            If UCase(strName) = UCase(udtVar.Name) Then
               If bytSkipThisVar = 1 Then
 ű
                 If udtSkipVar.Index <> udtVar.Index Then
 Q١
 U1
                   UniqueName = False
200
                   Exit For
 ij,
                 End If
               Else
                 UniqueName = False
                 Exit For
               End If
            End If
          Next udtVar
30
        End Function
        'Check enabled variables in collection for duplicate names.
        Public Function DuplicateNames() As Boolean
          Dim udtVar1 As Variable
          Dim udtVar2 As Variable
35
          Dim intI1 As Integer
          Dim intI2 As Integer
          DuplicateNames = False
```

NextID = lngID

```
For intI1 = 1 To mcolVariable.Count
            For intI2 = 1 To mcolVariable.Count
              If intI1 ⇔ intI2 Then
                 Set udtVar1 = mcolVariable.Item(intI1)
                 Set udtVar2 = mcolVariable.Item(intI2)
 5
                If udtVar1.Enabled And udtVar2.Enabled Then
                   If udtVar1.Name = udtVar2.Name Then
                     DuplicateNames = True
                     Exit Function
                   End If
10
                End If
              End If
            Next intI2
         Next intI1
15
       End Function
       Public Sub ReadCollection(ByVal strFN As String, ByVal lngStartIndex As Long, _
          ByVal lngEndIndex As Long)
 ď]
 đì
          mudtFile.FileName = strFN
          Call mudtFile.ReadFile(Me, lngStartIndex, lngEndIndex)
201
 ij
       End Sub
       Public Sub ReadObjects()
         Dim udtVar As Variable
          Dim udtType As VariableType
          On Error GoTo BeatIt
         Do Until Err.Number <> 0
30
            Set udtVar = New Variable
            udtType = udtVar.ReadType(mudtFile)
            Select Case udtType
              Case vtInteger
35
                 Set udtVar = New VarInteger
                 udtVar.Typ = vtInteger
              Case vtReal
                 Set udtVar = New VarReal
40
                 udtVar.Typ = vtReal
              Case vtFraction
```

```
Set udtVar = New VarFraction
                  udtVar.Typ = vtFraction
               Case vtString
                  Set udtVar = New VarString
                  udtVar.Typ = vtString
 5
               Case vtUntyped
                  Set udtVar = New VarUntyped
                  udtVar.Typ = vtUntyped
             End Select
10
             Call udtVar.ReadObjectData(mudtFile)
             udtVar.Index = NextID
             Call mcolVariable.Add(udtVar, Str(udtVar.Index))
15
          Loop
        BeatIt:
             Exit Sub
2Q)
        End Sub
 Q'1
 Me, die Me, der Mer, Mer
        Public Function WriteCollection(ByVal strFN As String, _
          ByVal lngIndexPos As Long, ByVal lngSeekPos) As Long
          mudtFile.FileName = strFN
          WriteCollection = mudtFile.WriteFile(Me, False, lngIndexPos, lngSeekPos)
25 T
          mblnIsDirty = False
        End Function
 <u>L</u>
        Public Sub WriteObjects()
30
          Dim udtVar As Variable
          For Each udtVar In mcolVariable
             Call udtVar.WriteObjectData(mudtFile)
          Next udtVar
35
        End Sub
        Public Sub Clear()
          'empties the collection class
```

```
Set mcolVariable = New Collection
       End Sub
       ' returns a collection of variables sorted by length of variable name,
 5
       'longest to shortest
       Public Function SortVarNamesByLength() As CVariables
          Dim udtVar As Variable
          Dim intLen As Integer
10
          Dim intLongest As Integer
          Dim udtCVar As New CVariables
          'Find longest variable name
          For Each udtVar In mcolVariable
            If udtVar.Enabled Then
15
               intLen = Len(udtVar.Name)
              If intLen > intLongest Then
                intLongest = intLen
 113
 Ōì
              End If
20
            End If
 Next udtVar
          'Sort variables by length of name - longest first
          Do
            For Each udtVar In mcolVariable
              If udtVar.Enabled Then
                 intLen = Len(udtVar.Name)
                 If intLen = intLongest Then
                   'Put this var in sorted collection
E)
                   udtCVar.AddObject udtVar
30
                 End If
               End If
            Next udtVar
            intLongest = intLongest - 1
          Loop While intLongest > 0
35
          Set SortVarNamesByLength = udtCVar
```

Set mcolVariable = Nothing

End Function

	VERSION 1.0 CLASS
	BEGIN
	MultiUse = -1 'True
5	END
	Attribute VB_Name = "CVariants"
	Attribute VB_GlobalNameSpace = False
	Attribute VB_Creatable = True
	Attribute VB_PredeclaredId = False
10	Attribute VB_Exposed = False
	Option Explicit
	'to hold collection
	Private mcolVariants As Collection
#19 #4 . #1	Private Sub Class_Initialize()
1 5	'creates the collection when this class is created
15 m u u ca u u ca	Set mcolVariant = New Collection
11	End Sub
10 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	
4]	
	Private Sub Class_Terminate()
200	ld-strong llostion value this close is tormainated
201	'destroys collection when this class is terminated Set mcolVariant = Nothing
þá en	Set meer arrant - rouning
L! Et	End Sub
= 2	
	Public Property Get Item(vntIndexKey As Variant) As Variant
25	'used when referencing an element in the collection
	'vntIndexKey contains either the Index or Key to the collection
	'this is why it is declared as a Variant
	'Syntax: Set foo = x.Item(xyz) or Set foo = x.Item(5)
20	Set Item = mcolVariant(vntIndexKey)
30	End Property
	Lite I topolity
	Public Property Get Count() As Long

'collection. Syntax: Debug.Print x.Count Count = mcolVariant.Count **End Property** Public Sub AddObject(udtVar As Variant) ' adds variable objects directly to the collection udtVar.Index = NextIDCall mcolVariant.Add(udtVar, Str(udtVar.Index)) End Sub Public Function Add(ByVal strName As String, _ ByVal strFrom As String, ByVal strTo As String, ByVal strBy As String) As Variant 'create a new object Dim udtVar As Variant Dim udtVarInteger As New VarInteger Set udtVar = udtVarInteger'set the properties passed into the method With udtVar .Name = strName .Index = NextIDEnd With With udtVarInteger .From = strFrom .Too = strTo.By = strByEnd With ' add the new object to the collection Call mcolVariant.Add(udtVarInteger, Str(udtVar.Index)) 'return the object created Set AddInteger = udtVarInteger

'used when retrieving the number of elements in the

5

10

Mind the Car

2₫*

30

35

End Function

Public Sub Remove(vntIndexKey As Variant)

'used when removing an element from the collection
'vntIndexKey contains either the Index or Key, which is why
'it is declared as a Variant
'Syntax: x.Remove(xyz)
mcolVariant.Remove vntIndexKey

End Sub

5

20¹

Public Property Get NewEnum() As IUnknown

'this property allows you to enumerate
'this collection with the For...Each syntax
Set NewEnum = mcolVariant.[_NewEnum]

End Property

Private Function NextID() As Long

' creates a unique index to associate a variable and the variable listbox Static lngID As Long

lngID = lngID + 1NextID = lngID

End Function

Public Sub Clear()

'empties the collection class

25 Set mcolVariant = Nothing Set mcolVariant = New Collection

End Sub

	' DifficultyEstimate.cls VERSION 1.0 CLASS
	BEGIN
5	MultiUse = -1 'True END
	Attribute VB_Name = "DifficultyEstimate"
	Attribute VB_GlobalNameSpace = False
	Attribute VB_Creatable = True Attribute VB PredeclaredId = False
10	Attribute VB_Exposed = False
	Option Explicit
	Private mblnIsDirty As Boolean
	Private Sub Class_Initialize()
1.5	mblnIsDirty = False
195	End Sub
11. 131	
Lī Fa	Public Property Let IsDirty(ByVal blnNewValue As Boolean)
1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	mblnIsDirty = blnNewValue
20	End Property
M. T.	Public Property Get IsDirty() As Boolean
	IsDirty = mblnIsDirty
C C	End Property
	'implemented in the subclasses of DifficultyEstimate
25	Public Function ComputeDifficulty() As Double
	End Function
	' implemented in the subclasses of DifficultyEstimate Public Function Copy() As DifficultyEstimate
	End Function
30	' implemented in the subclasses of DifficultyEstimate Public Sub ReadObjectData(udtFile As File)

End Sub

' implemented in the subclasses of DifficultyEstimate Public Sub WriteObjectData(udtFile As File)

End Sub

```
'DocStatus.cls
       VERSION 1.0 CLASS
       BEGIN
        MultiUse = -1 'True
 5
       END
       Attribute VB Name = "DocStatus"
       Attribute VB_GlobalNameSpace = False
       Attribute VB_Creatable = True
       Attribute VB PredeclaredId = False
10
       Attribute VB Exposed = False
       Option Explicit
       ' returns true if this document strFN is open
       Public Function IsOpen(ByVal strFN As String) As Boolean
         Dim docD As Document
15
         For Each docD In Documents
            If InStr(1, strFN, docD.Name) Then
 Ų)
              IsOpen = True
 đì
              Exit Function
Ų"
            End If
20.
 IJ.
         Next docD
         IsOpen = False
 End Function
```

	'DSMODEL.CLS
	VERSION 1.0 CLASS
	BEGIN
	MultiUse = -1 'True
5	Persistable = 0 'NotPersistable
	DataBindingBehavior = 0 'vbNone
	DataSourceBehavior = 0 'vbNone
	MTSTransactionMode = 0 'NotAnMTSObject
	END
10	Attribute VB Name = "DSModel"
	Attribute VB GlobalNameSpace = False
	Attribute VB Creatable = False
	Attribute VB PredeclaredId = False
	Attribute VB Exposed = False
15	Option Explicit
	Implements Model
	Dim mudtModel As Model
n M	
M. 65.	Private Sub Class_Initialize()
1. 1 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.	Set mudtModel = New Model
	End Sub
	' Delegated to Class Model
H.	Public Property Get Model FileName() As String
C)	Tublic Troporty Got Model_Thertaine() 715 bumg
	Model_FileName = mudtModel.FileName
an L	End Property
25	' Delegated to Class Model
25	Public Property Let Model_FileName(ByVal strNewValue As String)
	mudtModel.FileName = strNewValue
	End Property
	' Delegated to Class Model
30	Public Property Get Model_IsFrozen() As Boolean
	Model_IsFrozen = mudtModel.IsFrozen

		End Property
ı		' Delegated to Class Model Public Property Let Model_IsFrozen(ByVal blnNewValue As Boolean)
		mudtModel.IsFrozen = blnNewValue
,	5	End Property
		' Delegated to Class Model Public Property Get Model_Comments() As String
)		Model_Comments = mudtModel.Comments
		End Property
)	10	'Delegated to Class Model Public Property Let Model_Comments(ByVal strNewValue As String)
		mudtModel.Comments = strNewValue
	the test	End Property
١	1.	' Delegated to Class Model Public Property Get Model_Clones() As CClones
		Set Model_Clones = mudtModel.Clones
,		End Property
		' Delegated to Class Model Public Property Get Model_Variables() As CVariables
,	20	Set Model_Variables = mudtModel.Variables
		End Property
ł		' Delegated to Class Model Public Property Get Model_Constraints() As CConstraints
		Set Model_Constraints = mudtModel.Constraints
	25	End Property

'Delegated to Class Model

	Public Sub Model_AddChecksum(ByVal dblChecksum As Double)
	Call mudtModel.AddChecksum(dblChecksum)
	End Sub
5	' Delegated to Class Model Public Sub Model_InitChecksums()
	mudtModel.InitChecksums
	End Sub
	' Delegated to Class Model Public Sub Model_InitTempChecksums()
10	mudtModel.InitTempChecksums
	End Sub
in Harr And An Town I was Said	'Delegated to Class Model Public Function Model_ChecksumExists(ByVal dblChecksum As Double) As Boolean
	Model_ChecksumExists = mudtModel.ChecksumExists(dblChecksum)
15	End Function
1	' Delegated to Class Model Public Property Let Model_IsDirty(ByVal blnNewValue As Boolean)
	mudtModel.IsDirty = blnNewValue
L.	End Property
20	' Delegated to Class Model Public Property Get Model_IsDirty() As Boolean
	Model_IsDirty = mudtModel.IsDirty
	End Property
25	' Delegated to Class Model Public Property Let Model_LastClone(ByVal intNewValue As Integer)
	mudtModel.LastClone = intNewValue

End Property
' Delegated to Class Model Public Property Get Model_LastClone() As Integer
Model_LastClone = mudtModel.LastClone
End Property
' Delegated to Class Model Public Sub Model_FreezeModel()
Call mudtModel.FreezeModel
End Sub
' Delegated to Class Model Public Sub Model_OpenDoc(ByVal udtWord As MSWord)
Call mudtModel.OpenDoc(udtWord)
End Sub
' Delegated to Class Model Public Sub Model_CloseDoc()
Call mudtModel.CloseDoc
End Sub
' Delegated to Class Model Public Sub Model_CloseAllCloneDocs()
Call mudtModel.CloseAllCloneDocs
End Sub
' Delegated to Class Model Public Sub Model_ReadModel()
mudtModel.ReadModel

25 End Sub

5

10

The state of the s

20

' Delegated to Class Model

```
Public Sub Model ReadObjects()
         mudtModel.ReadObjects
       End Sub
       ' Delegated to Class Model
       Public Sub Model WriteModel()
 5
         mudtModel.WriteModel
       End Sub
       ' Delegated to Class Model
       Public Sub Model WriteObjects()
10
         mudtModel.WriteObjects
       End Sub
       'Delegated to Class Model
       Public Function Model ConstraintsOK(ByVal udtTestType As TestType, _
         ByVal udtProlog As Prolog, blnUnderconstrained As Boolean,
15]
         blnTestAborted As Boolean, strUnderconstrainedVN As String) As Boolean
 ij,
         Model_ConstraintsOK = mudtModel.ConstraintsOK(udtTestType, udtProlog, _
            blnUnderconstrained, blnTestAborted, strUnderconstrainedVN)
       End Function
       'implemented here
       Public Sub Model GenerateClones(ByVal udtWord As MSWord, ByVal udtProlog As Prolog, _
         ByVal intNumClones As Integer, ByVal bytDifference As Byte)
         Call mudtModel.SubstituteValues(Me, udtWord, udtProlog, intNumClones,
            bytDifference, 285)
       End Sub
25
       ' Delegated to Class Model
       Public Sub Model SubstituteValues(ByVal objO As Object,
         ByVal udtWord As MSWord, ByVal udtProlog As Prolog,
         ByVal intNumClones As Integer, ByVal bytDifference As Byte,
         ByVal intStartPos As Integer)
```

30

End Sub

```
Public Sub CreateVariant(ByVal udtClone As Clone)
          Dim rnumber As Integer
          Dim statementRange As Range
 5
          Dim firstNSE As String
          Dim secondNSE As String
          With udtClone.CloneDoc
            rnumber = .Tables(1).Rows.Count * Rnd + 0.5
             .Tables(1).Cell(Row:=rnumber, Column:=1).Range.Copy
             firstNSE = .Tables(1).Cell(Row:=rnumber, Column:=2).Range.Text
10
             firstNSE = left(firstNSE, 1)
            Set statementRange = .Bookmarks("tca fStatement").Range
            statementRange.Paste
             .Tables(1).ConvertToText
15
             .Bookmarks.Add name:="tca_fStatement", Range:=statementRange
            statementRange.Borders.OutsideLineStyle = wdLineStyleSingle
            ' trim hard returns at end of statement
            Dim i, n As Integer
            Dim retchr As String
20
            retchr = Chr\$(13)
            With statementRange
 Ú
               n = 0
               i = .Words.Count
 E)
               While .Words(i).Text = retchr And i > 1
25
                i = i - 1
                If .Words(i).Text = retchr Then
                  n = n + 1
                End If
               Wend
30
               If n > 0 Then
                 .Words(.Words.Count - n + 1).Delete Count:=n
               End If
            End With
            rnumber = .Tables(2).Rows.Count * Rnd + 0.5
            .Tables(2).Cell(Row:=rnumber, Column:=1).Range.Copy
35
```

```
secondNSE = .Tables(2).Cell(Row:=rnumber, Column:=2).Range.Text
            secondNSE = left(secondNSE, 1)
            Set statementRange = .Bookmarks("tca sStatement").Range
            statementRange.Paste
 5
            .Tables(1).ConvertToText
            .Bookmarks.Add name:="tca_sStatement", Range:=statementRange
            statementRange.Borders.OutsideLineStyle = wdLineStyleSingle
            ' trim hard returns at end of statement
            With statementRange
10
              n = 0
              i = .Words.Count
               While .Words(i).Text = retchr And i > 1
                i = i - 1
                If .Words(i).Text = retchr Then
15
                  n = n + 1
                End If
               Wend
 ij,
              If n > 0 Then
                 . Words (. Words . Count - n + 1). Delete Count := n + 1
20]
              End If
            End With
 ď.
            Dim key As String
            Dim keyChr As String
            If firstNSE = "N" And secondNSE = "N" Then
              kev = "E"
            ElseIf firstNSE = "S" And secondNSE = "S" Then
              key = "C or E"
            ElseIf firstNSE = "E" And secondNSE = "E" Then
              kev = "D"
30
            ElseIf firstNSE = "N" And secondNSE = "S" Then
              key = "E"
            ElseIf firstNSE = "E" And secondNSE = "S" Then
              key = "A"
            ElseIf firstNSE = "S" And secondNSE = "E" Then
35
              kev = "B"
            ElseIf firstNSE = "N" And secondNSE = "E" Then
              kev = "B"
            ElseIf firstNSE = "E" And secondNSE = "N" Then
```

key = "A"

```
End If
            keyChr = left(.Bookmarks("key").Range.Text, 1)
            If keyChr = "A" Or keyChr = "1" Then
             key = "A"
            ElseIf keyChr = "B" Or keyChr = "2" Then
 5
            ElseIf keyChr = "C" Or keyChr = "3" Then
             key = "C"
            ElseIf keyChr = "D" Or keyChr = "4" Then
10
            ElseIf keyChr = "E" Or keyChr = "5" Then
             key = "E"
            End If
            Dim keyRange As Range
15
            Set keyRange = .Bookmarks("tca_Key").Range
            If key = "" Then
              keyRange.InsertBefore Text:="TCA cannot determine the key"
            Else
              keyRange.InsertBefore Text:="Key is " & key
            End If
            udtClone.key = key
         End With
       End Sub
```

	' Family.cls VERSION 1.0 CLASS
	BEGIN MultiUse = -1 'True
5 10	END Attribute VB_Name = "Family" Attribute VB_GlobalNameSpace = False Attribute VB_Creatable = True Attribute VB_PredeclaredId = False Attribute VB_Exposed = False
	Option Explicit
	' current version of data produced by this class Const mintVERSIONSTAMP As Integer = 1
15	' enable i/o Private mudtFile As File
	' the .mdf file name of this family Private mstrFamilyFN As String
900 GT, 1910 II. 10 10 10 10 10 10	' the program that owns this family Private mudtProgram As Program
	' the item type Private mudtItemType As ItemType
	' close/medium far classification Private mudtProximity As Proximity
C C 25	' generic/non-generic classification Private mblnGeneric As Boolean
	' accession number, if this family is based on a locked item Private mstrAccNum As String
	' the active model Private mudtActiveModel As Model
30	' collection of Models Private mudtCModels As CModels
	' the collection of accepted clones Private mudtCClones As CClones

```
' is dirty?
       Private mblnIsDirty As Boolean
       Public Enum Program
         prGRE = 0
 5
         prGMAT = 1
         prSAT = 2
         prMR = 3
       End Enum
       Public Enum ItemType
         ptStandardMC = 0
10
         ptQuantComp = 1
         ptDataSuff = 2
       End Enum
       Public Enum Proximity
15
         prNear = 0
         prMedium = 1
         prFar = 2
 ď1
       End Enum
 G1
 IJ
       Private Enum FamilyRecordLayout
201
         frLocalDataIndex = 1 ' long (takes 4 bytes)
         frCloneIndex = 5 long
         frLocalData = 51
         frClones = 201 'variable length
       End Enum
       Private Sub Class_Initialize()
         Set mudtCModels = New CModels
         Set mudtCClones = New CClones
         mblnIsDirty = False
       End Sub
30
       Public Property Get FileName() As String
         FileName = mstrFamilyFN
       End Property
       Public Property Let FileName(ByVal strNewValue As String)
```

mstrFamilyFN = left(strNewValue, Len(strNewValue) - 4) & ".mdf" **End Property** Public Property Get Program() As Program 5 Program = mudtProgram **End Property** Public Property Let Program(ByVal udtNewValue As Program) mudtProgram = udtNewValue 10 **End Property** Public Property Get ItemType() As ItemType ItemType = mudtItemType ű, 15 **End Property** 20 Public Property Let ItemType(ByVal udtNewValue As ItemType) mudtItemType = udtNewValue **End Property** Public Property Get Proximity() As Proximity Proximity = mudtProximity **End Property** Public Property Let Proximity(ByVal udtNewValue As Proximity) mudtProximity = udtNewValue 25 **End Property** Public Property Get Generic() As Boolean Generic = mblnGeneric **End Property**

	Public Property Let Generic(ByVal blnNewValue As Boolean)
	mblnGeneric = blnNewValue
	End Property
5	Public Property Get AccNum() As String
3	AccNum = mstrAccNum
	End Property
	Public Property Let AccNum(ByVal strNewValue As String)
	mstrAccNum = strNewValue
10	End Property
<u> </u>	Public Property Get ActiveModel() As Model
	Set ActiveModel = mudtActiveModel
15	End Property
15	Public Property Let ActiveModel(ByVal udtModel As Model)
## care	Set mudtActiveModel = udtModel
	End Property
	Public Property Get Models() As CModels
20	Set Models = mudtCModels
	End Property
	Public Property Get Clones() As CClones
	Set Clones = mudtCClones
	End Property
25	Public Property Let IsDirty(ByVal blnNewValue As Boolean)
	mblnIsDirty = blnNewValue

```
End Property
       Private Property Get IsDirty() As Boolean
         If mudtCClones.IsDirty Or mblnIsDirty Then
 5
            IsDirty = True
          Else
            IsDirty = False
          End If
10
       End Property
       Public Sub CloseAllCloneDocs()
         Dim udtClone As Clone
          For Each udtClone In mudtCClones
            udtClone.CloseDoc
 ű,
          Next udtClone
       End Sub
       Public Sub ReadFamily()
         Dim udtWAPI As New Win32API
         If udtWAPI.FileExists(mstrFamilyFN) Then
            Set mudtFile = New File
            mudtFile.FileName = mstrFamilyFN
            Call mudtFile.ReadFile(Me, frLocalDataIndex, frCloneIndex)
251
            Set mudtFile = Nothing
            Call mudtCClones.ReadCollection(mstrFamilyFN, frCloneIndex, READ_UNTIL_EOF)
         End If
30
       End Sub
       Public Sub ReadObjects()
         Dim vField As Variant
         Call mudtFile.ReadField(vField) ' returns the version stamp
35
         Call mudtFile.ReadField(vField)
         Program = vField
         Call mudtFile.ReadField(vField)
```

5	ItemType = vField Call mudtFile.ReadField(vField) Generic = vField Call mudtFile.ReadField(vField) Proximity = vField Call mudtFile.ReadField(vField) AccNum = vField
	End Sub
10	Public Sub WriteFamily()
	Dim udtPB As New Progress
15 III "::" III O "	If IsDirty Then Set mudtFile = New File mudtFile.FileName = mstrFamilyFN Call udtPB.Init(2, "Saving family") Call mudtFile.WriteFile(Me, True, frLocalDataIndex, frLocalData) udtPB.Advance Set mudtFile = Nothing Call mudtCClones.WriteCollection(mstrFamilyFN, frCloneIndex, frClones) udtPB.Advance End If
2 5	IsDirty = False
	End Sub
	Public Sub WriteObjects()
L.	Call mudtFile.WriteField(mintVERSIONSTAMP)
	Call mudtFile.WriteField(Program)
30	Call mudtFile.WriteField(ItemType)
	Call mudtFile.WriteField(Generic)
	Call mudtFile.WriteField(Proximity)
	Call mudtFile WriteField(AccNum)

End Sub

```
'File.cls
        VERSION 1.0 CLASS
        BEGIN
         MultiUse = 0 'False
 5
         Persistable = 0 'NotPersistable
         DataBindingBehavior = 0 'vbNone
         DataSourceBehavior = 0 'vbNone
         MTSTransactionMode = 0 'NotAnMTSObject
        END
        Attribute VB Name = "File"
10
        Attribute VB GlobalNameSpace = False
        Attribute VB Creatable = True
        Attribute VB PredeclaredId = False
        Attribute VB Exposed = False
        Option Explicit
15
        ' Path and name of the file to open
        Private m sFileName As String
 ų,
20-5
        'File number opened
        Private m iFileNumber As Integer
        'passed in by ReadFile
        Private mlngEndPos As Long
        'Error constants
        Enum FileError
 ď,
          fileOpenError = vbObjectError + 512 + 2
          fileEOFError = vbObjectError + 512 + 3
25
 C
          fileReadError = vbObjectError + 512 + 4
          fileWriteError = vbObjectError + 512 + 5
          fileStopReadingError = vbObjectError + 512 + 6
        End Enum
30
        Property Get FileName() As String
        Attribute FileName.VB Description = "Name of the file to contain the task information."
          FileName = m sFileName
        End Property
35
        Property Let FileName(ByVal sFileName As String)
          'Should validate valid path here
```

```
End Property
       'Reads all objects from a file into the defined object
 5
       ' Parameters:
       Public Sub ReadFile(obj As Object, Optional ByVal lngStartIndex As Long = 0,
          Optional ByVal lngEndIndex As Long = 0)
          Dim lngStartPos As Long
10
          ' Enable error handling
          On Error Resume Next
          'Get the file number
          m iFileNumber = FreeFile
15
          'Open the file and trap any errors
          Open m sFileName For Binary Access Read As #m iFileNumber
 ű
          Select Case err. Number
 Øì
20
            Case 0 'No error
 Ų,
               If lngEndIndex > 0 Then
                 Seek m iFileNumber, lngEndIndex
                 Get #m_iFileNumber, , mlngEndPos
               Else
                 mlngEndPos = 0
              End If
              If lngStartIndex > 0 Then
 þ
                 Seek m iFileNumber, lngStartIndex
                 Get #m iFileNumber, , lngStartPos
301
                 Seek m iFileNumber, lngStartPos
               End If
               obj.ReadObjects 'Get the data
            Case 53 'File not found
35
               ' Do nothing
            Case Else
               'Turn off error handling here
               On Error GoTo 0
40
               ' Pass the error out
               err.Raise fileOpenError, "CFile::ReadFile", "Error opening file."
```

m sFileName = sFileName

```
End Select
          'Close the file
 5
          Close #m iFileNumber
        End Sub
        'Reads a field from the file
        ' Parameters:
        ' vField
                   field read from the file
10
        Public Sub ReadField(vField As Variant)
          ' Set the error handler
          On Error GoTo ERR_HANDLER
15
          Get #m iFileNumber, , vField
 ű
          If EOF(m iFileNumber) Then
            'Reached end of file
 Ui
            err.Raise fileEOFError
26"
          End If
          If mlngEndPos > 0 Then
            If mlngEndPos < Seek(m iFileNumber) Then
               err.Raise fileStopReadingError
            End If
          End If
 le i
 C)
        Exit Sub
        ERR HANDLER:
30
          ' Pass the error out
          Select Case err. Number
            Case fileEOFError
               Call err.Raise(err.Number, "File::ReadField", "EOF")
            Case fileStopReadingError
35
               Call err.Raise(err.Number, "File::ReadField", "Stop!")
            Case Else
               Call err.Raise(fileReadError, "File::ReadField", err.Description)
```

40

End Select

End Sub

```
' Writes all objects to the file.
        ' Parameters:
        ' obj
                  Object
 5
        Public Function WriteFile(obj As Object, _
          Optional ByVal blnKillOldFile As Boolean = False,
          Optional ByVal lngIndexPos As Long = 0, _
          Optional ByVal lngSeekPos As Long = 1) As Long
          'Enable error handling
10
          On Error Resume Next
          If blnKillOldFile Then 'assume new file, otherwise append
            Kill m sFileName 'Kill the existing file
            err.Clear
          End If
          'Get the file number
          m_iFileNumber = FreeFile
          'Open the file and trap any errors
          Open m sFileName For Binary As #m iFileNumber
 ij,
20=
           'write the starting file position, if lngIndexPos > 0
 ű
          If lngIndexPos > 0 Then
            Seek m iFileNumber, lngIndexPos
            Put #m iFileNumber, , lngSeekPos
          End If
          ' seek to starting position
          Seek m iFileNumber, lngSeekPos
          Select Case err. Number
30
            Case 0 'No error
               ' Write the data
               obj.WriteObjects
            Case Else
               'Turn off error handling here
35
               On Error GoTo 0
               ' Pass the error out
               err.Raise fileOpenError, "CFile::WriteFile",
                 "Error opening file: " & err.Description
```

End Select

'return current position WriteFile = Seek(m_iFileNumber) 5

> 'Close the file Close #m_iFileNumber

- 10 **End Function**
 - 'Write a field to the file
 - ' Parameters:
 - ' vField field to write to the file

Public Sub WriteField(ByVal vField As Variant)

'Set the error handler 15 On Error GoTo ERR_HANDLER

Put #m_iFileNumber, , vField

Exit Sub ERR_HANDLER: err.Raise fileWriteError, "CFile::WriteField", _ "Write Error: " & err.Descpription

End Sub

ij,

(T

Uī

20:

Han Bun Han

	'FileFind.cls VERSION 1.0 CLASS BEGIN
5	MultiUse = -1 'True
5	END Attribute VB Name = "FileFind"
	Attribute VB_Name = Ther had Attribute VB GlobalNameSpace = False
	Attribute VB Creatable = True
	Attribute VB_PredeclaredId = False
10	Attribute VB_Exposed = False
	Option Explicit
	'used for finding files that fit a mask
	Private Type FILETIME
15	dwLowDateTime As Long
	dwHighDateTime As Long
	End Type
# % E	Private Const MAX_PATH = 260
	Private Type WIN32_FIND_DATA
20-	dwFileAttributes As Long
	ftCreationTime As FILETIME
	ftLastAccessTime As FILETIME
	ftLastWriteTime As FILETIME nFileSizeHigh As Long
25	nFileSizeLow As Long
Pi.	dwReserved0 As Long
	dwReserved1 As Long
#=#	cFileName As String * MAX_PATH
	cAlternate As String * 14
30	End Type
	Private Const INVALID_HANDLE_VALUE = -1
	Private Declare Function FindFirstFile Lib "kernel32" Alias "FindFirstFileA" _ (ByVal lpFileName As String, lpFindFileData As WIN32_FIND_DATA) As Long
35	Private Declare Function FindNextFile Lib "kernel32" Alias "FindNextFileA" _ (ByVal hFileName As Long, lpFindFileData As WIN32_FIND_DATA) As Long
	Private Declare Function FindClose Lib "kernel32" (ByVal hFindFile As Long) As Long

	Private Declare Function GetCurrentDirectory Lib "kernel32" _ Alias "GetCurrentDirectoryA" (ByVal nBufferLength As Long, _ ByVal lpBuffer As String) As Long
5	' returns true if strFN exists Public Function Exists(ByVal strFN) As Boolean
	Dim lngHandle As Long Dim w32FindData As WIN32_FIND_DATA
10	lngHandle = FindFirstFile(strFN, w32FindData)
10	If lngHandle = INVALID_HANDLE_VALUE Then Exists = False
15	Else Exists = True Call FindClose(IngHandle) End If
	End Function
1 5 C C C C C C C C C C C C C C C C C C	'returns a collection of file names that satisfy strMask. The path seems to 'disappear from the returned file names.
2 0	Public Function FindAll(ByVal strMask As String) As Collection
25 L	Dim lngHandle As Long Dim lngRet As Long Dim w32FindData As WIN32_FIND_DATA Dim strFN As String Dim varI As Variant Dim colFNs As New Collection
	lngHandle = FindFirstFile(strMask, w32FindData)
30	If lngHandle = INVALID_HANDLE_VALUE Then Exit Function End If
35	Do varI = InStr(1, w32FindData.cFileName, Chr(0)) ' trim off the nulls strFN = left(w32FindData.cFileName, varI - 1) Call colFNs.Add(strFN) ' add to the collection
	Loop Until FindNextFile(lngHandle, w32FindData) = 0

Set FindAll = colFNs

End Function

5 'returns the current directory
Public Function CurrentDirectory() As String

Dim strBuf As String Dim lngRet As Long Dim varI As Variant

10

strBuf = Space(300) lngRet = GetCurrentDirectory(300, strBuf) varI = InStr(1, strBuf, Chr(0)) ' trim off the nulls CurrentDirectory = left(strBuf, varI - 1)

15

Then offer offer offen them that then

The first state of

End Function

	GWAT Difficulty Estimate. Of S
	VERSION 1.0 CLASS
	BEGIN
	MultiUse = -1 'True
5	
5	END
	Attribute VB_Name = "GMATDifficultyEstimate"
	Attribute VB_GlobalNameSpace = False
	Attribute VB Creatable = True
	Attribute VB PredeclaredId = False
10	Attribute VB Exposed = False
10	Option Explicit
	Option Explicit
	Layrent version of data produced by this class
	'current version of data produced by this class
	Const mintVERSIONSTAMP As Integer = 1
	Lucian anta Difficulta Estimata
	Implements DifficultyEstimate
1.5	Drivete mudtDE As DifficultyEstimate
15	Private mudtDE As DifficultyEstimate
43	1.1
۵ì	' these go into the GMAT model
in	Private mudtDomain As Domain
jr Ma	Private mstrKey As String
ú1	Private mudtNature As Nature
20:	Private mudtItemType As ItemType
113	Private mintTDDiffEst As Integer
	1 1 1 W V 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
	Private Sub Class Initialize()
£	1 Tivate Sub Class_Hittatize()
Ē	9-4 44DF - N D: 65 14 E-4
E.	Set mudtDE = New DifficultyEstimate
** ***********************************	F 10.1
2 5]	End Sub
	Private Sub Class_Terminate()
	Set mudtDE = Nothing
	End Sub
	Public Property Get DifficultyEstimate_IsDirty() As Boolean
30	DifficultyEstimate_IsDirty = mudtDE.IsDirty
-	
	End Property

```
Public Property Let DifficultyEstimate IsDirty(ByVal blnNewValue As Boolean)
                                   mudtDE.IsDirty = blnNewValue
                           End Property
    5
                           Public Property Let Domain(ByVal udtNewValue As Domain)
                                   mudtDomain = udtNewValue
                           End Property
                           Public Property Let Nature(ByVal udtNewValue As Nature)
10
                                   mudtNature = udtNewValue
                           End Property
                          Public Property Let Key(ByVal strNewValue As String)
mstrKey = strNewValue
                           End Property
                           Public Property Let ItemType(ByVal udtNewValue As ItemType)
                                   mudtItemType = udtNewValue
    The street of th
                           End Property
                           Public Property Let TDDiffEst(ByVal intNewValue As Integer)
                                  mintTDDiffEst = intNewValue
20
                           End Property
                          Public Function DifficultyEstimate ComputeDifficulty() As Double
                                  Dim dblDiff As Double
                                  dblDiff = -2.3289902
25
                                   ' add coeff for domain
                                  If mudtDomain = doAlgebra Then
                                          dblDiff = dblDiff + 0.2341578
```

```
ElseIf mudtDomain = doGeometry Then
             dblDiff = dblDiff + 0.3749013
          End If
          ' add coeff for real
 5
          If mudtNature = naReal Then
             dblDiff = dblDiff + 0.3285613
          End If
10
          ' add coeff for td difficulty estimate
          dblDiff = dblDiff + ((6 - mintTDDiffEst) * 0.7024191)
          ' add coeff for key
          If mudtItemType = ptDataSuff Then
             If mstrKey = "A" Or mstrKey = "B" Then
15
               dblDiff = dblDiff + 0.7334054
             End If
          End If
20)
01
          DifficultyEstimate ComputeDifficulty = dblDiff
        End Function
 Ų1
 Mil. Mr. Mil. Mil.
        ' returns a copy of this object
        Public Function DifficultyEstimate Copy() As DifficultyEstimate
25
[]
          Dim udtGmatDE As New GMATDifficultyEstimate
          Set DifficultyEstimate Copy = udtGmatDE
        End Function
        Public Sub DifficultyEstimate ReadObjectData(udtFile As File)
          Dim vField As Variant
30
          Call udtFile.ReadField(vField) ' reads the version stamp
        End Sub
35
        Public Sub DifficultyEstimate WriteObjectData(udtFile As File)
          Call udtFile.WriteField(mintVERSIONSTAMP)
          mudtDE.IsDirty = False
```

	GREDIfficulty Estimate.cis
	VERSION 1.0 CLASS
	BEGIN
	MultiUse = -1 'True
5	END
-	Attribute VB_Name = "GREDifficultyEstimate"
	Attribute VB_GlobalNameSpace = False
	Attribute VB Creatable = True
	-
10	Attribute VB_PredeclaredId = False
10	Attribute VB_Exposed = False
	Option Explicit
	' current version of data produced by this class
	Const mintVERSIONSTAMP As Integer = 1
	Implements DifficultyEstimate
15,	Private mudtDE As DifficultyEstimate
₩# .#4	
14.6 171	' these go into the GRE model
₩: 111	Private mudtDomain As Domain
# 1	Private mudtComputation As GREComputation
177	Private mudtCognition As GRECognition
15 % % % % % % % % % % % % % % % % % % %	Private mudtConcept As GREConcept
153 - 4	Private mstrKey As String
	Private mudtNature As Nature
	Private mudtItemType As ItemType
47	111vate maditemit ype 115 itemi ype
an fluid "30" fluid	Public Enum GREComputation
25	grIntegers = 0
2 4]	grDecimalsFractions = 1
C)	grRadicals = 2
	grNone = 3
	End Enum
30	Public Enum GRECognition
	grProcedural = 0
	grConceptual = 1
	grHigherOrderThinking = 2
	End Enum
35	Public Enum GREConcept
33	grProbability = 0
	grPercentofPercent = 1
	D

```
grPercentChange = 2
         grLinearInequality = 3
         grNoneOfThese = 4
       End Enum
 5
       Private Sub Class_Initialize()
         Set mudtDE = New DifficultyEstimate
       End Sub
       Private Sub Class_Terminate()
10
         Set mudtDE = Nothing
       End Sub
       Public Property Get DifficultyEstimate_IsDirty() As Boolean
 T.
         DifficultyEstimate IsDirty = mudtDE.IsDirty
01
U1
15)
       End Property
       Public Property Let DifficultyEstimate IsDirty(ByVal blnNewValue As Boolean)
         mudtDE.IsDirty = blnNewValue
       End Property
       Public Property Let Domain(ByVal udtNewValue As Domain)
         mudtDomain = udtNewValue
       End Property
       Public Property Get Computation() As GREComputation
         Computation = mudtComputation
25
       End Property
       Public Property Let Computation(ByVal udtNewValue As GREComputation)
         mudtComputation = udtNewValue
```

```
mudtDE.IsDirty = True
          End If
       End Property
 5
       Public Property Get Cognition() As GRECognition
          Cognition = mudtCognition
       End Property
       Public Property Let Cognition(ByVal udtNewValue As GRECognition)
          If mudtCognition 	<> udtNewValue Then
10
            mudtCognition = udtNewValue
            mudtDE.IsDirty = True
          End If
       End Property
       Public Property Get Concept() As GREConcept
 <u>a</u>1
 U
15.
          Concept = mudtConcept
 ű
       End Property
       Public Property Let Concept(ByVal udtNewValue As GREConcept)
          If mudtConcept 	<> udtNewValue Then
            mudtConcept = udtNewValue
            mudtDE.IsDirty = True
         End If
       End Property
       Public Property Get Nature() As Nature
          Nature = mudtNature
25
       End Property
       Public Property Let Nature(ByVal udtNewValue As Nature)
          mudtNature = udtNewValue
```

```
End Property
        Public Property Get Key() As String
          Key = mstrKey
        End Property
        Public Property Let Key(ByVal strNewValue As String)
          If mstrKey <> strNewValue Then
            mstrKey = strNewValue
            mudtDE.IsDirty = True
          End If
10
        End Property
        Public Property Get ItemType() As ItemType
          ItemType = mudtItemType
 41
 Man Alle Bon Ham band
        End Property
        Public Property Let ItemType(ByVal udtNewValue As ItemType)
15
          mudtItemType = udtNewValue
        End Property
        Public Function DifficultyEstimate ComputeDifficulty() As Double
          Dim dblDiff As Double
20
          dblDiff = 0.3296816
          ' add coeff for domain
          If mudtDomain = doAlgebra Then
            dblDiff = dblDiff + 0.2464302
25
          ElseIf mudtDomain = doDataAnalysis Then
            dblDiff = dblDiff - 0.3944198
          End If
          ' add coeff for computation
          If mudtComputation = grIntegers Then
30
            dblDiff = dblDiff - 0.8563799
```

```
ElseIf mudtComputation = grDecimalsFractions Then
            dblDiff = dblDiff - 0.5181709
          End If
 5
          ' add coeff for cognition
          If mudtCognition = grProcedural Then
            dblDiff = dblDiff - 0.6621277
            If mudtNature = naReal Then 'add coeff for procedural and real
               dblDiff = dblDiff - 0.8781659
10
            End If
          ElseIf mudtCognition = grHigherOrderThinking Then
            dblDiff = dblDiff + 0.7253093
          End If
          ' add coeff for concept
15
          Select Case mudtConcept
            Case grLinearInequality
               dblDiff = dblDiff - 0.5881492
            Case grNoneOfThese
               ' do nothing
            Case Else
               dblDiff = dblDiff + 0.5835095
 Mir. Mars Mars
          End Select
25.
          ' add coeff for key
          If mudtItemType = ptQuantComp Then
            If mstrKey = "A" Or mstrKey = "B" Or mstrKey = "C" Then
               dblDiff = dblDiff - 0.531099
            End If
          End If
          DifficultyEstimate ComputeDifficulty = dblDiff
       End Function
35
       ' returns a copy of this object
       Public Function DifficultyEstimate Copy() As DifficultyEstimate
          Dim udtGreDE As New GREDifficultyEstimate
          udtGreDE.Computation = Computation
          udtGreDE.Cognition = Cognition
          udtGreDE.Concept = Concept
40
          Set DifficultyEstimate Copy = udtGreDE
```

Public Sub DifficultyEstimate ReadObjectData(udtFile As File) Dim vField As Variant 5 Call udtFile.ReadField(vField) ' reads the version stamp Call udtFile.ReadField(vField) Computation = vField Call udtFile.ReadField(vField) Cognition = vField 10 Call udtFile.ReadField(vField) Concept = vFieldEnd Sub Public Sub DifficultyEstimate WriteObjectData(udtFile As File) Ū Call udtFile.WriteField(mintVERSIONSTAMP) Call udtFile.WriteField(Computation) Call udtFile.WriteField(Cognition) 20% Call udtFile.WriteField(Concept) mudtDE.IsDirty = False Ü End Sub ₽å 25 'IniFile.cls **VERSION 1.0 CLASS BEGIN** MultiUse = -1 'True **END** 30 Attribute VB Name = "IniFile" Attribute VB GlobalNameSpace = False Attribute VB Creatable = True Attribute VB PredeclaredId = False Attribute VB Exposed = False 'this class handles all ini file reads and writes via kernel32

End Function

35

Option Explicit ' the following declares are needed to get and put to .ini files Private Declare Function GetPrivateProfileSection Lib "kernel32" Alias "GetPrivateProfileSectionA" (ByVal lpAppName As String, _ ByVal lpReturnedString As String, ByVal nSize As Long, _ ByVal lpFileName As String) As Long Private Declare Function GetPrivateProfileString Lib "kernel32" Alias "GetPrivateProfileStringA" (ByVal lpApplicationName As String, _ ByVal lpKeyName As Any, ByVal lpDefault As String, ByVal lpReturnedString As String, ByVal nSize As Long, _ ByVal lpFileName As String) As Long Private Declare Function WritePrivateProfileSection Lib "kernel32" Alias "WritePrivateProfileSectionA" (ByVal lpAppName As String, _ ByVal lpString As String, ByVal lpFileName As String) As Long Private Declare Function WritePrivateProfileString Lib "kernel32" Alias "WritePrivateProfileStringA" (ByVal lpApplicationName As String, _ ByVal lpKeyName As Any, ByVal lpString As Any, ByVal lpFileName As String) As Long ' contains file name of ini Private mstrFN As String 'holds collection of keys created by Get ProfileSection method Private mcolKeys As Collection 'holds collection of values created by Get ProfileSection method Private mcolValues As Collection Private Sub Class Initialize() Set mcolKeys = New Collection Set mcolValues = New Collection End Sub ' sets the ini path + file name Public Property Let FN(ByVal strFN As String)

5

10

15

20

Hen alben flete effen street

30

35

mstrFN = strFN

End Property

J

```
' returns the ini path + file name
        Public Property Get FN() As String
           FN = mstrFN
 5
        End Property
        'gets all of the keys and values in a section
        Public Sub GetProfileSection(ByVal strSectionName As String)
           Dim strRet As String
           strRet = Space(5000)
           If GetPrivateProfileSection(strSectionName, strRet, 5000, mstrFN) = 0 Then
10
             Call MsgBox("Ini file call unsuccessful", vbExclamation, "Error")
           End If
           Dim lngStart As Long
           Dim lngEnd As Long
15
           Dim str1 As String
 ű
           Dim str2 As String
 g1
 Mrs. day Am
           Dim varT As Variant
           Dim strT As String
20
           ' parse the key and variable names out of strRet, add to the collections
           For lngStart = 1 To Len(strRet)
             str1 = Mid(strRet, lngStart, 1)
             If str1 \Leftrightarrow Chr(0) Then
                For lngEnd = lngStart + 1 To Len(strRet)
                  str2 = Mid(strRet, lngEnd, 1)
                  Select Case str2
                     Case "="
                       strT = Mid(strRet, lngStart, lngEnd - lngStart)
                       Call mcolKeys.Add(strT)
30
                       Exit For
                     Case Chr(0)
                       strT = Mid(strRet, lngStart, lngEnd - lngStart)
                       Call mcolValues.Add(strT)
                     Exit For
                  End Select
35
                Next lngEnd
                lngStart = lngEnd
             End If
           Next lngStart
```

40

End Sub

Dim intl As Integer

```
'called after LoadProfileSection.
        'sets strKey and strValue to the KeyValue pairs if one exists
        ' at this index.
       'returns TRUE if the index exists, FALSE if it doesn't.
 5
        Public Function GetKeyValuePair(strKey As String, strValue As String, _
          ByVal intIndex As Integer) As Boolean
          If intIndex <= mcolKeys.Count Then
            strKey = mcolKeys.Item(intIndex)
            strValue = mcolValues.Item(intIndex)
10
            GetKeyValuePair = True
          Else
            strKey = ""
            strValue = ""
            GetKeyValuePair = False
15
          End If
 IJ
        End Function
 Œ1
 U
        ' init before loading key/value pairs
       Public Function InitializeKeyValuePairs()
20
 ű
          Set mcolKeys = Nothing
          Set mcolValues = Nothing
          Set mcolKeys = New Collection
          Set mcolValues = New Collection
        End Function
       Public Sub SetKeyValuePair(ByVal strKey As String, ByVal strValue As String)
          Call mcolKeys.Add(strKey)
          Call mcolValues.Add(strValue)
       End Sub
30
       Public Sub WriteProfileSection(ByVal strSectionName As String)
          Dim strSection As String
          Dim varKey As Variant
          Dim varValue As Variant
```

	For Each varKey In mcolKeys intI = intI + 1 varValue = mcolValues.Item(intI) strSection = strSection & varKey & "=" & varValue & Chr(0)
5	Next varKey
	If WritePrivateProfileSection(strSectionName, strSection, mstrFN) = 0 Then Call MsgBox("Ini file write section call unsuccessful", _ vbExclamation, "Error")
10	End If
	End Sub
	' returns the number of keys currently in the key/value collections Public Property Get NumKeys() As Integer
15	NumKeys = mcolKeys.Count
	End Property
uj Ol	'gets a value
070 2944	Public Function GetProfileString(ByVal strSectionName As String, _ ByVal strKeyName As String) As String
	Dim strRet As String strRet = Space(5000)
• C. T. C.	Call GetPrivateProfileString(strSectionName, strKeyName, "Not Found", _ strRet, 5000, mstrFN)
2 <u>5.</u>	GetProfileString = TrimAtFirstNull(strRet)
LJ Z	End Function
	'sets a value Public Sub WriteProfileString(ByVal strSectionName As String, _ ByVal strKeyName As String, ByVal strKeyValue As String)
30	Call WritePrivateProfileString(strSectionName, strKeyName, strKeyValue, mstrFN)
	End Sub

	'LockedItem.cls
	VERSION 1.0 CLASS
	BEGIN MultiUse = -1 'True
5	END
3	Attribute VB Name = "LockedItem"
	Attribute VB_GlobalNameSpace = False
	Attribute VB Creatable = True
	Attribute VB PredeclaredId = False
10	Attribute VB_Exposed = False
	Option Explicit
	Private mstrLockedFN As String
	Private mudtWord As MSWord
	Private mdocLockedItem As Document
	Private mudtItemType As ItemType
1	Private mudtDeliveryMode As DeliveryMode
7] 71	Public Enum DeliveryMode
## ###	dmCBT = 0
W.	dmPPT = 1
2 <u>0</u>	End Enum
20 # C # C	Public Property Let LockedItemFileName(ByVal strNewValue As String)
	mstrLockedFN = strNewValue
incod	End Property
	Public Property Let WordInstance(ByVal udtNewValue As MSWord)
25	Set mudtWord = udtNewValue
	End Property
	Public Property Get DeliveryMode() As DeliveryMode
	DeliveryMode = mudtDeliveryMode
	End Property

```
Public Property Get ItemType() As ItemType
         ItemType = mudtItemType
       End Property
       Public Function OpenLockedItemDoc() As Boolean
         Dim udtProgress As New Progress
         Call udtProgress.Init(2, "Opening locked item...")
         udtProgress.Advance
         Set mdocLockedItem = mudtWord.WordApp.Documents.Open(mstrLockedFN)
         If mdocLockedItem.ProtectionType <> wdNoProtection Then
            Call mdocLockedItem.Unprotect("ItemEdit")
10
         End If
         OpenLockedItemDoc = AnalyzeLockedItem
Le Karten Kar
         udtProgress.Advance
       End Function
       Public Sub CloseLockedItemDoc()
         mdocLockedItem.Close
         Clipboard.Clear
       End Sub
       Private Function AnalyzeLockedItem() As Boolean
20
         ' true if document is successfully analyzed
         AnalyzeLockedItem = True
         If mdocLockedItem.Tables.Count = 1 Then 'QC item
            mudtItemType = ptQuantComp
           If mdocLockedItem.Bookmarks.Count = 3 Then
25
              mudtDeliveryMode = dmPPT
           Else
              mudtDeliveryMode = dmCBT
           End If
```

	ElseIf mdocLockedItem.ListParagraphs.Count = 2 Then 'DS mudtItemType = ptDataSuff mudtDeliveryMode = dmCBT
5	ElseIf mdocLockedItem.ListParagraphs.Count = 5 Then 'SMC mudtItemType = ptStandardMC mudtDeliveryMode = dmCBT
	ElseIf mdocLockedItem.Bookmarks.Exists("prop_key") = True Then 'SMC mudtItemType = ptStandardMC mudtDeliveryMode = dmPPT
10	Else AnalyzeLockedItem = False End If
	End Function
6]	Public Sub ConvertCBTSMCItem()
1 <u>5</u> 1	Dim udtProgress As New Progress
## ## ##	Call udtProgress.Init(2, "Converting SMC CBT locked item")
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	Dim tcaDoc As Document Set tcaDoc = mudtWord.WordApp.ActiveDocument
20 444	Dim stemRange As Range Set stemRange = mdocLockedItem.Content stemRange.Find.Style = "Heading 2" stemRange.Find.Execute FindText:="Stem" stemRange.Start = stemRange.Start + 5
25	Dim respRange As Range Set respRange = mdocLockedItem.Content respRange.Find.Style = "Heading 2" respRange.Find.Execute FindText:="Response"
	stemRange.End = respRange.Start - 1 stemRange.Copy
30	Dim destRange As Range Set destRange = tcaDoc.Bookmarks("stem1").Range
	With destRange

```
.Borders.Enable = False
            .Words(1).Delete Count:=6
            .Collapse
            .Paste
 5
            .Style = wdStyleNormal
            .Borders.Enable = True
          End With
          destRange.Borders.Enable = False
          destRange.Collapse
          destRange.Delete
10
          destRange.Paste
          destRange.InsertParagraphAfter
          destRange.Style = wdStyleNormal
          destRange.Borders.Enable = True
15
          With destRange.ParagraphFormat.Borders
           .Enable = True
           .DistanceFromTop = 1
           .DistanceFromLeft = 4
 ű
           .DistanceFromBottom = 1
 (I)
           .DistanceFromRight = 4
          End With
 Min Ann Min Bur
          If destRange.Borders.InsideLineStyle = True Then
           destRange.Borders.InsideLineStyle = wdLineStyleNone
          End If
         tcaDoc.Bookmarks.Add Name:="stem1", Range:=destRange
         Dim nextRange As Range
         Dim Key As String
         Dim abcde As String
          abcde = "ABCDE"
30
          Dim i As Integer
         Dim n As Integer
         n = 1
         Dim udtIF As New IniFile
         udtIF.FN = IN DIRECTORY & ExtractFileNameNoExt(mstrLockedFN) & ".ini"
35
         Key = udtIF.GetProfileString("LockedItemData", "Key")
         udtProgress.Advance
```

Dim tabchr As String

```
tabchr = Chr\$(9)
          For i = 1 To 5
            Set respRange = mdocLockedItem.ListParagraphs(i).Range
           respRange.Copy
            If Key = Mid(abcde, i, 1) Then
 5
              Set destRange = tcaDoc.Bookmarks("Key").Range
            Else
              Set destRange = tcaDoc.Bookmarks("resp" & Format(n)).Range
              n = n + 1
            End If
10
            With destRange
              .Borders.Enable = False
              .Words(1).Delete
              .Collapse
              .Paste
              .Style = wdStyleNormal
              .Borders.Enable = True
              If .Words(1).Text = tabchr Then
                .Words(1).Delete
2₫=
              End If
              . Words(destRange. Words. Count). Delete
           End With
          Next
          udtProgress.Advance
25
       End Sub
       Public Sub ConvertPPTSMCItem()
          Dim udtProgress As New Progress
          Call udtProgress.Init(2, "Converting SMC PPT locked item...")
          Dim tcaDoc As Document
          Set tcaDoc = mudtWord.WordApp.ActiveDocument
30
          Dim stemStart As Long
```

	Dim destRange As Range Set destRange = tcaDoc.Bookmarks("stem1").Range stemStart = destRange.Start
5	Dim stemRange As Range Set stemRange = mdocLockedItem.Bookmarks("itemnum").Range stemRange.Start = stemRange.Start + 1 Set stemRange = mdocLockedItem.Content stemRange.Find.Style = "PPTStimulus"
10	If stemRange.Find.Execute Then stemRange.Copy destRange.Paste destRange.Collapse Direction:=wdCollapseEnd End If
15 D. T. C. T.	Set stemRange = mdocLockedItem.Content stemRange.Find.Style = "PPTStem" stemRange.Find.Execute stemRange.Copy destRange.Paste destRange.Style = wdStyleNormal
	destRange.Start = stemStart destRange.Borders.Enable = True
	DistanceFromTop = 1 DistanceFromLeft = 4 DistanceFromBottom = 1
30	If destRange.Borders.InsideLineStyle = True Then destRange.Borders.InsideLineStyle = wdLineStyleNone End If
	tcaDoc.Bookmarks.Add Name:="stem1", Range:=destRange
35	Dim nextRange As Range Dim respRange As Range Dim Key As String Dim abcde As String abcde = "ABCDE"

```
Dim i As Integer
         Dim n As Integer
          n = 1
         Dim udtIF As New IniFile
 5
          udtIF.FN = IN DIRECTORY & ExtractFileNameNoExt(mstrLockedFN) & ".ini"
          Key = udtIF.GetProfileString("LockedItemData", "Key")
          udtProgress.Advance
         For i = 1 To 5
           Set respRange = mdocLockedItem.Content
           respRange.Find.Style = "PPTOptions"
10
           respRange.Find.Execute FindText:="(" & Mid(abcde, i, 1) & ")"
           respRange.Start = respRange.Start + 4
           Set nextRange = mdocLockedItem.Content
           If i < 5 Then
              nextRange.Find.Style = "PPTOptions"
              nextRange.Find.Execute FindText:="(" & Mid(abcde, i + 1, 1) & ")"
           Else
              nextRange.Find.Style = "ItemLabel"
              nextRange.Find.Execute FindText:="Scratch Pad"
201
           End If
           respRange.End = nextRange.Start - 1
           respRange.Copy
 ļ-h
           If Key = Mid(abcde, i, 1) Then
              Set destRange = tcaDoc.Bookmarks("Key").Range
              Set destRange = tcaDoc.Bookmarks("resp" & Format(n)).Range
              n = n + 1
           End If
           destRange.Words(1).Delete
           destRange.Collapse
30
           destRange.Paste
          Next
          udtProgress.Advance
```

Public Sub ConvertDSItem() Dim udtProgress As New Progress Call udtProgress.Init(2, "Converting DS CBT locked item...") 5 Dim tcaDoc As Document Set tcaDoc = mudtWord.WordApp.ActiveDocument Dim stemRange As Range Set stemRange = mdocLockedItem.Content stemRange.Find.Style = "Heading 2" 10 stemRange.Find.Execute FindText:="Stem" stemRange.Start = stemRange.Start + 5Dim respRange As Range Set respRange = mdocLockedItem.Content respRange.Find.Style = "DataSuffStatement" respRange.Find.Execute 157 Min. Man. Man. April 47. stemRange.End = respRange.Start - 1stemRange.Copy Dim destRange As Range Set destRange = tcaDoc.Bookmarks("stem1").Range destRange.Borders.Enable = False destRange.Collapse destRange.Paste destRange.Borders.Enable = True With destRange.ParagraphFormat.Borders .Enable = True25 .DistanceFromTop = 1.DistanceFromLeft = 4 .DistanceFromBottom = 1 .DistanceFromRight = 4 End With 30 If destRange.Borders.HasHorizontal = True Then destRange.Borders(wdBorderHorizontal).LineStyle = wdLineStyleNone End If

End Sub

Dim Key As String

	Dim udtIF As New IniFile udtIF.FN = IN_DIRECTORY & ExtractFileNameNoExt(mstrLockedFN) & ".ini" Key = udtIF.GetProfileString("LockedItemData", "Key")
5	Set destRange = tcaDoc.Bookmarks("Key").Range destRange.Words(1).Delete destRange.InsertBefore Text:=Key
	udtProgress.Advance
	Dim i As Integer
	For i = 1 To 2
10	Set respRange = mdocLockedItem.ListParagraphs(i).Range respRange.Copy
Harry 6" Reads famp	Set destRange = tcaDoc.Tables(i).Cell(Row:=1, Column:=1).Range destRange.Paste destRange.Style = wdStyleNormal
1 5 1	Next
17. 17. 17. 17. 17. 17. 17. 17. 17. 17.	udtProgress.Advance
	End Sub
11 11 12 12 12 12 13 13 13 13 13 13 13 13 13 13 13 13 13	Public Sub ConvertCBTQCItem()
	Dim udtProgress As New Progress
20)	Call udtProgress.Init(2, "Converting QC CBT locked item")
	Dim tcaDoc As Document Set tcaDoc = mudtWord.WordApp.ActiveDocument
25	Dim stemRange As Range Set stemRange = mdocLockedItem.Tables(1).Cell(Row:=1, Column:=1).Range stemRange.Copy
20	Dim destRange As Range Set destRange = tcaDoc.Bookmarks("stem1").Range destRange.Borders.Enable = False destRange.Words(2).Delete
30	destRange.Words(1).Delete

5	destRange.Collapse destRange.Paste tcaDoc.Tables(2).Rows.SetLeftIndent LeftIndent:=-0.6, RulerStyle:=wdAdjustNone tcaDoc.Tables(2).ConvertToText Separator:=wdSeparateByTabs destRange.Borders.Enable = True
3	tcaDoc.Bookmarks.Add Name:="stem1", Range:=destRange
10	Dim Key As String Dim udtIF As New IniFile udtIF.FN = IN_DIRECTORY & ExtractFileNameNoExt(mstrLockedFN) & ".ini" Key = udtIF.GetProfileString("LockedItemData", "Key")
	Set destRange = tcaDoc.Bookmarks("Key").Range destRange.Words(1).Delete destRange.InsertBefore Text:=Key
	udtProgress.Advance
	Dim respRange As Range Set respRange = mdocLockedItem.Tables(1).Cell(Row:=2, Column:=1).Range respRange.Copy Set destRange = tcaDoc.Bookmarks("columnA").Range destRange.Collapse destRange.Paste
	Set respRange = mdocLockedItem.Tables(1).Cell(Row:=2, Column:=2).Range respRange.Copy Set destRange = tcaDoc.Bookmarks("columnB").Range destRange.Collapse destRange.Paste
	udtProgress.Advance
	End Sub
	Public Sub ConvertPPTQCItem()
	Dim udtProgress As New Progress
30	Call udtProgress.Init(2, "Converting QC PPT locked item")
	Dim tcaDoc As Document Set tcaDoc = mudtWord.WordApp.ActiveDocument
	Dim stemRange As Range

	Dim destRange As Range Set destRange = tcaDoc.Bookmarks("stem1").Range
5	destRange.Borders.Enable = False destRange.Words(2).Delete
	destRange. Words(1). Delete
	destRange.Collapse
0	destRange.Paste tcaDoc.Tables(2).Rows.SetLeftIndent LeftIndent:=-0.6, RulerStyle:=wdAdjustNon
	tcaDoc.Tables(2).ConvertToText Separator:=wdSeparateByTabs
	destRange.Borders.Enable = True tcaDoc.Bookmarks.Add Name:="stem1", Range:=destRange
	teaboc.bookmarks.Add Name.— stemi , Kange.—destikange
_	Dim Key As String
5	Dim udtIF As New IniFile udtIF.FN = IN DIRECTORY & ExtractFileNameNoExt(mstrLockedFN) & ".ini"
E 1	Key = udtIF.GetProfileString("LockedItemData", "Key")
	Set destRange = tcaDoc.Bookmarks("Key").Range
# T	destRange. Words(1). Delete
	destRange.InsertBefore Text:=Key
16° () 28 () 28 () 28 () 28 () 28. ()	udtProgress.Advance
= F 1	Dim respRange As Range
	Set respRange = mdocLockedItem.Tables(1).Cell(Row:=2, Column:=2).Range respRange.Copy
5 <u>.</u>	Set destRange = tcaDoc.Bookmarks("columnA").Range
Cj	destRange.Collapse
<u>C</u> j	destRange.Paste
	Set respRange = mdocLockedItem.Tables(1).Cell(Row:=2, Column:=4).Range
80	respRange.Copy Set destRange = tcaDoc.Bookmarks("columnB").Range
	destRange.Collapse
	destRange.Paste
	udtProgress.Advance

 $Set\ stemRange = mdocLockedItem. Tables (1). Cell (Row:=1,\ Column:=2). Range$

stemRange.Copy

End Sub

VBSCA -361-

	Model.cis
	VERSION 1.0 CLASS
	BEGIN
	MultiUse = -1 'True
5	Persistable = 0 'NotPersistable
	DataBindingBehavior = 0 'vbNone
	DataSourceBehavior = 0 'vbNone
	MTSTransactionMode = 0 'NotAnMTSObject
	END
10	Attribute VB Name = "Model"
	Attribute VB GlobalNameSpace = False
	Attribute VB Creatable = True
	Attribute VB PredeclaredId = False
	Attribute VB Exposed = False
15	Attribute VB Ext KEY = "SavedWithClassBuilder", "Yes
	Attribute VB Ext KEY = "Top Level", "No"
	Option Explicit
C 1	
er Li	'current version of data produced by this class
G1	Const mintVERSIONSTAMP As Integer = 1
Li1	
20=	' enable i/o
43	Private mudtFile As File
	'handle for Model
e Eî	Private mdocModel As Document
ui:	141 - 1 - 61 641 1 - 1
ر آ	' the .doc file name of this model
	Private mstrDocFN As String
	' the .mdl file name of this model
	Private mstrConFN As String
	' has this model produced variants that were accepted?
	Private mblnIsFrozen As Boolean
	Tivate momisi 102011 713 Doolean
30	' comments about this model
	Private mstrComments As String
	2
	' all of the variables for this model
	Private mudtCVariables As CVariables
	' all of the constraints for this model
35	Private mudtCConstraints As CConstraints

' all of the clones generated by this model Private mudtCClones As CClones ' the collection of checksums accepted by this model (these persist) Private mcolChecksums As Collection ' the collection of checksums accepted by this model (these don't persist) Private mcolTempChecksums As Collection ' the Prolog object Private mudtProlog As Prolog ' needed for I/O Private mblnProcessChecksums As Boolean ' is dirty? Private mblnIsDirty As Boolean 'needed to save the model one last time after it's frozen Private mblnFreeze As Boolean ű, Ō٦ 15 Private Enum ModelRecordLayout Mrs. Ann Mrs. Ann mrLocalDataIndex = 1 ' long (takes 4 bytes) mrVariableIndex = 5 ' longmrConstraintIndex = 9 ' longmrChecksumIndex = 13 ' 'long mrLocalData = 51 ' byte mrVariables = 201 'variable length ' the constraint data starts wherever the checksum data ends ' the checksum data starts wherever the constraint data ends End Enum Private Sub Class Initialize() Set mudtCVariables = New CVariables Set mudtCConstraints = New CConstraints Set mudtCClones = New CClones Set mcolChecksums = New Collection Set mcolTempChecksums = New Collection mblnIsDirty = True mblnFreeze = FalseEnd Sub

Public Property Get FileName() As String

5

10

30

```
FileName = mstrDocFN
       End Property
       Public Property Let FileName(ByVal strNewValue As String)
          mstrDocFN = strNewValue
          ' create the FN for the constraint file
          mstrConFN = left(mstrDocFN, Len(mstrDocFN) - 4) & ".mdl"
       End Property
       Public Property Get IsFrozen() As Boolean
          IsFrozen = mblnIsFrozen
10
       End Property
       Public Property Let IsFrozen(ByVal blnNewValue As Boolean)
 Per Alea Mer Aley Ann
          mblnIsFrozen = blnNewValue
       End Property
       Public Property Get Comments() As String
          Comments = mstrComments
       End Property
       Public Property Let Comments(ByVal strNewValue As String)
          If mstrComments <> strNewValue Then
            mstrComments = strNewValue
20
            mblnIsDirty = True
          End If
       End Property
       Public Property Get Clones() As CClones
          Set Clones = mudtCClones
25
       End Property
```

```
Public Property Get Variables() As CVariables
          Set Variables = mudtCVariables
        End Property
        Public Property Get Constraints() As CConstraints
          Set Constraints = mudtCConstraints
 5
        End Property
        Public Sub FreezeModel()
          If IsFrozen = False Then
            mblnFreeze = True
10
            IsFrozen = True
            WriteModel
          End If
        End Sub
 Men allen f
        Public Sub AddChecksum(ByVal dblChecksum As Double)
15
          Call mcolChecksums.Add(dblChecksum)
          mblnIsDirty = True
 C
        End Sub
        ' resets the checksums if this model is a child
        Public Sub InitChecksums()
          Set mcolChecksums = New Collection
20
        End Sub
        Private Sub AddTempChecksum(ByVal dblChecksum As Double)
          Call mcolTempChecksums.Add(dblChecksum)
        End Sub
25
        ' resets the temp checksums if this model is changed and variants are deleted
        Public Sub InitTempChecksums()
```

Set mcolTempChecksums = New Collection

End Sub

٥'n

The Branks has

20

ij,

Public Function ChecksumExists(ByVal dblChecksum As Double) As Boolean

Dim vntChecksum As Variant

' if no variables were checksummed, consider the variant unique
If dblChecksum = 0 Then
ChecksumExists = False
Exit Function
End If

'check the persistent checksums (from accepted or discarded variants)

For Each vntChecksum In mcolChecksums

If vntChecksum = dblChecksum Then

ChecksumExists = True

Exit Function

End If

End If Next vntChecksum

' check the checksums of variants produced in this session
For Each vntChecksum In mcolTempChecksums
If vntChecksum = dblChecksum Then
ChecksumExists = True
Exit Function
End If
Next vntChecksum

ChecksumExists = False

End Function

Public Property Let IsDirty(ByVal blnNewValue As Boolean)

mblnIsDirty = blnNewValue

End Property

Public Property Get IsDirty() As Boolean

30 Dim mblnSaved As Boolean

^{&#}x27; As frozen models never get saved, they report is dirty

```
' when they are read in from disk. This fix causes them
          ' to always report not IsDirty.
          If IsFrozen Then
            IsDirty = False
            Exit Property
 5
          End If
          If mdocModel Is Nothing Then
            mblnSaved = True
          Else
            mblnSaved = mdocModel.Saved
10
          End If
          If mblnIsDirty Or
            mudtCVariables.IsDirty Or
            mudtCConstraints.IsDirty Or
15
            mblnSaved = False Then
              IsDirty = True
          Else
            IsDirty = False
          End If
 đ١
 U1
20
       End Property
 ű,
       Public Property Let LastClone(ByVal intNewValue As Integer)
          mudtCClones.SeqNum = intNewValue
       End Property
       Public Property Get LastClone() As Integer
          LastClone = mudtCClones.SeqNum
       End Property
       ' displays model
       Public Sub OpenDoc(ByVal udtWord As MSWord)
          Dim udtDS As New DocStatus
30 .
          ' see if word doc is open
          If udtDS.IsOpen(mstrDocFN) = False Then
            Set mdocModel = udtWord.WordApp.Documents.Open(mstrDocFN, , mblnIsFrozen)
          End If
```

mdocModel.Activate

	End Sub
	' closes model Public Sub CloseDoc()
5	' save the model and the word doc Call WriteModel
	Dim udtDS As New DocStatus
10	' close the word doc If udtDS.IsOpen(mstrDocFN) Then Call mdocModel.Close(False) ' don't save Set mdocModel = Nothing End If
	End Sub
	Public Sub CloseAllCloneDocs()
15 41	Dim udtClone As Clone
15 mm of the second of the sec	For Each udtClone In mudtCClones udtClone.CloseDoc Next udtClone
n C. w. L.	End Sub
2 0 5	Public Sub ReadModel()
le#	Dim udtWAPI As New Win32API
25	If udtWAPI.FileExists(mstrConFN) Then Set mudtFile = New File mudtFile.FileName = mstrConFN mblnProcessChecksums = False Call mudtFile.ReadFile(Me, mrLocalDataIndex, mrVariableIndex) Call mudtCVariables.ReadCollection(mstrConFN, mrVariableIndex, mrConstraintIndex) Call mudtCConstraints.ReadCollection(mstrConFN, mrConstraintIndex,
30	mrChecksumIndex) mblnProcessChecksums = True Call mudtFile.ReadFile(Me, mrChecksumIndex, READ_UNTIL_EOF) Set mudtFile = Nothing

```
End If
        End Sub
        Public Sub ReadObjects()
          Dim vField As Variant
          If mblnProcessChecksums Then
 5
            On Error GoTo BeatIt
            Do Until err.Number ⇔ 0
               Call mudtFile.ReadField(vField)
               Call mcolChecksums.Add(vField)
10
            Loop
          Else
            Call mudtFile.ReadField(vField) ' returns the version stamp
            Call mudtFile.ReadField(vField)
            LastClone = vField
            Call mudtFile.ReadField(vField)
            IsFrozen = vField
            Call mudtFile.ReadField(vField)
            Comments = vField
          End If
  ij
       BeatIt:
  ų)
          Exit Sub
       End Sub
       Public Sub WriteModel()
          Dim IngEndPos As Long
25
          Dim udtDS As New DocStatus
          Dim udtProg As New Progress
          If IsDirty = False Then Exit Sub
          If IsFrozen And mblnFreeze = False Then Exit Sub
          Call udtProg.Init(2, "Saving the active model...")
          If udtDS.IsOpen(mstrDocFN) Then ' see if word doc is open
30
            If Not IsFrozen Then 'command will fail if doc is read-only
               mdocModel.Save
            End If
```

	End If Set mudtFile = New File
	mudtFile.FileName = mstrConFN
_	mblnProcessChecksums = False
5	Call mudtFile.WriteFile(Me, True, mrLocalDataIndex, mrLocalData) udtProg.Advance
	lngEndPos = mudtCVariables.WriteCollection(mstrConFN, mrVariableIndex, mrVariables) lngEndPos = mudtCConstraints.WriteCollection(mstrConFN, mrConstraintIndex, lngEndPos) mblnProcessChecksums = True
10	Call mudtFile.WriteFile(Me, False, mrChecksumIndex, lngEndPos) Set mudtFile = Nothing
	udtProg.Advance
	IsDirty = False
	mblnFreeze = False
15	End Sub
æ	Public Sub WriteObjects()
130112045 2045	Dim vntChecksum As Variant
Ÿ1	If mblnProcessChecksums Then
42 2011	For Each vntChecksum In mcolChecksums
20## ###	Call mudtFile.WriteField(vntChecksum) Next vntChecksum
	Else
2 424	Call mudtFile.WriteField(mintVERSIONSTAMP)
111 111	Call mudtFile.WriteField(LastClone)
2 5	Call mudtFile.WriteField(IsFrozen)
<u>ļ</u>	Call mudtFile.WriteField(Comments)
" L L L L	End If
.	End Sub
	' tests the constraints, doesn't care about unique solution
30	Public Function ConstraintsOK(ByVal udtTestType As TestType,
	ByVal udtProlog As Prolog, blnUnderconstrained As Boolean, _
	blnTestAborted As Boolean, strUnderconstrainedVN As String) As Boolean
	Dim strVN As String
	Dim strVal As String
35	Dim udtCS As ConstraintSolver
	Set udtCS = InitConstraintSolver(2, udtTestType)
	VBSCA -370-

```
udtCS.Prolog = udtProlog
         blnUnderconstrained = False
 5
         blnTestAborted = False
         Select Case udtCS.Solve(srTest)
            Case srPrologError, srNoSolutions
              ConstraintsOK = False
10
              Exit Function
            Case srPrologAborted
              blnTestAborted = True
              ConstraintsOK = False
              Exit Function
15
            Case srSuccess
              Do While udtCS.GetNextValue(strUnderconstrainedVN, strVal)
                If strVal = " " Then ' it's underconstrained
                   ConstraintsOK = False
                   blnUnderconstrained = True
                   Exit Function
200
                End If
 Loop
         End Select
         ConstraintsOK = True
       End Function
       ' implemented in the subclasses of Model
       Public Sub GenerateClones(ByVal udtWord As MSWord, ByVal udtProlog As Prolog, _
         ByVal intNumClones As Integer, ByVal bytDifference As Byte)
       End Sub
       'common code called by GenerateClones in the subclasses
       Public Sub SubstituteValues(ByVal objO As Object,
         ByVal udtWord As MSWord, ByVal udtProlog As Prolog,
35
         ByVal intNumClones As Integer, ByVal bytDifference As Byte,
         ByVal intStartPos As Integer)
         Dim udtClone As Clone
         Dim strPath As String
         Dim fRange As Range
40
         Dim intIndex As Integer
         Dim udtCS As ConstraintSolver
```

5		Dim udtSortedVs As CVariables Dim udtCon As Constraint Dim strVarName As String Dim strValue As String Dim intTry As Integer Dim blnSolFound As Boolean Dim blnUniqueSolFound As Boolean Dim udtType As VariableType
10		CloseDoc ' close the model doc CommandBars("File").Controls("Exit").Enabled = False Randomize
		' do substitution of values into model doc
		strPath = ExtractPath(FileName)
15 	1	Dim udtProgress As New Progress Call udtProgress.Init(intNumClones, "Generating variants")
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1		'initalize the constraint solver Set udtCS = InitConstraintSolver(bytDifference) udtCS.Prolog = udtProlog
20 ¹		' solve loop For intIndex = 1 To intNumClones
# T		' try 10x to get a unique sol, then give up
77. me		For intTry = 1 To 10 DoEvents ' allow abort
25		If frmProlog.Abort Then
- T== 		Exit Sub
		End If
		blnSolFound = False
30		blnUniqueSolFound = False If udtCS.Solve(srGenerate) Then 'found a variant
30		blnSolFound = True
		Else
		Exit For
25		End If
35		'variant found - is it unique? If Not ChecksumExists(udtCS.Checksum) Then blnUniqueSolFound = True Exit For
40		End If Next intTry
10		roat mary

```
'error if no solution found
            If Not blnSolFound Then
              Call MsgBox("No solution could be found for this constraint set", _
                 vbExclamation, "Error")
 5
                 udtProgress.Kill
              Exit Sub
            End If
            'error if unique solution could not be found
            If Not blnUniqueSolFound Then
              Call MsgBox("A unique solution could not be found for this constraint set after 10
10
       attempts." &
                 " You may want to try again.", vbExclamation, "Error")
                 udtProgress.Kill
              Exit Sub
            End If
15
            ' add the new clone to the collection
            Set udtClone = Clones.Add(ExtractFileName(FileName), True)
            udtClone.Checksum = udtCS.Checksum
            Call AddTempChecksum(udtClone.Checksum)
            ' add the new clone to the disposition list box
            With frmTCA.lstDisposition
              Call .AddItem(udtClone.FileName)
               .ItemData(.ListCount - 1) = udtClone.index
            End With
            FileCopy FileName, strPath & udtClone.FileName
            Call udtClone.OpenDoc(udtWord, strPath)
            ' do the substitution
            Set fRange = udtClone.CloneDoc.Content
            fRange.start = intStartPos
            With fRange.find
               While udtCS.GetNextValue(strVarName, strValue)
                 .ClearFormatting
                 .Text = strVarName
                 .Replacement.ClearFormatting
                 .Replacement.Text = FormatValue(strVarName, strValue)
35
                 'this first execute needed so Word returns correct value
                 .Execute replace:=wdReplaceAll, Forward:=True,
                   MatchCase:=True
               Wend
40
            End With
            Dim i, n As Integer
            Dim nShapes As Long
```

```
n = udtClone.CloneDoc.InlineShapes.Count
             For i = 1 To n
               udtCS.ResetValueIndex
 5
               While udtCS.GetNextValue(strVarName, strValue)
                 udtClone.CloneDoc.InlineShapes(i).Select
                 Call MTTextSubstitution(strVarName, strValue)
               Wend
             Next
10
            udtClone.CloneDoc.Bookmarks("stem1").Range.Copy
            If udtClone.CloneDoc.Bookmarks.Exists("tca Stem") = True Then
               Dim stemRange As Range
               Set stemRange = udtClone.CloneDoc.Bookmarks("tca Stem").Range
               stemRange.Paste
15
               udtClone.CloneDoc.Bookmarks.Add name:="tca Stem", Range:=stemRange
            Else
               Call MsgBox("Model is missing TCA Stem Bookmark!", vbExclamation, "Hey!")
  Ų.
            End If
  Ø1
  Uî
20
            ' trim hard returns at end of stem
  4)
            Dim retchr As String
            retchr = Chr\$(13)
            With stemRange
              n = 0
              i = .Words.Count
               While . Words(i). Text = retchr And i > 1 'Rob: I added the And part. Pete
                i = i - 1
                If .Words(i).Text = retchr Then
                  n = n + 1
                End If
30
               Wend
              If n > 0 Then
                 .Words(.Words.Count - n + 1).Delete Count:=n
              End If
            End With
35
            ' callback to subclass to code unique to this model type
            Call objO.CreateVariant(udtClone)
             udtProgress.Advance
```

```
End Sub
        ' create, initialize constraint solver
        Private Function InitConstraintSolver(ByVal bytDifference As Byte,
 5
          Optional ByVal udtTestType As TestType = tcTestAll) As ConstraintSolver
          Dim udtVar As Variable
          Dim udtCon As Constraint
          Dim udtVarString As VarString
          Dim udtCS As New ConstraintSolver
10
          Dim udtSortedVs As CVariables
          'add enabled variables to ConstraintSolver object, sorted by length,
          ' strings first
          Set udtSortedVs = mudtCVariables.SortVarNamesByLength
          For Each udtVar In udtSortedVs
            If udtVar.Enabled Then
              Call udtCS.AddVariable(udtVar)
            End If
          Next udtVar
          ' Add enabled constraints
          For Each udtCon In Constraints
            If udtCon.Enabled Then
              If udtTestType = tcTestAll Or
                 udtCon.ConstraintType = udtTestType - 1 Then
                 Call udtCS.AddConstraint(udtCon)
              End If
            End If
          Next udtCon
30
          udtCS.DiffWeight = bytDifference
          Set InitConstraintSolver = udtCS
        End Function
        ' formats all math variables for item presentation
        Private Function FormatValue(ByVal strVarName As String, _
```

ByVal strValue As String) As String

udtClone.CloseDoc

Next intIndex

35

```
Dim udtV As Variable
          Dim udtVR As VarReal
          Dim udtVF As VarFraction
          For Each udtV In mudtCVariables
 5
            If udtV.Enabled Then
               If udtV.name = ExtractVarName(strVarName) Then
                 Select Case udtV.Typ
                   Case vtInteger
                      FormatValue = strValue
                   Case vtReal
10
                      Set udtVR = udtV
                      FormatValue = FormatReal(strValue, _
                        udtVR.DecimalPlaces, udtVR.TrailingZeros)
                   Case vtFraction
                      Set udtVF = udtV
15
                     If udtVF.MixedNumbers Then
                        FormatValue = FormatFraction(strValue)
                     Else
                        FormatValue = strValue
                      End If
20
                   Case vtString
                     FormatValue = strValue
                   Case vtUntyped
                     FormatValue = FormatUntyped(strValue)
25
                 End Select
                 Exit For
              End If
            End If
          Next udtV
        End Function
        ' takes the index off of a string variable name that is indexed
        Private Function ExtractVarName(ByVal strName As String) As String
          Dim varI As Variant
          varI = InStr(1, strName, ".")
35
          If var I > 0 Then
            ExtractVarName = left(strName, varI - 1)
          Else
            ExtractVarName = strName
          End If
```

End Function

```
' formats reals for item presentation
        Private Function FormatReal(ByVal strReal As String, ByVal intPlaces As Integer, _
          ByVal blnTZeros As Boolean) As String
 5
          Dim varPos As Variant
          Dim intLen As Integer
          Dim strI As String
          Dim strD As String
          Dim blnZeroFound As Boolean
          varPos = InStr(1, strReal, ".")
10
          ' isolate strings on either side of decimal point
          If varPos = 0 Then
             strI = strReal
          Else
             strI = Mid(strReal, 1, varPos - 1)
             strD = Mid(strReal, varPos + 1, Len(strReal))
          End If
          intLen = Len(strD)
          ' pad or trim to intPlaces
          If intLen < intPlaces Then
             strD = strD & String(intPlaces - intLen, "0")
          Else
             If intLen > intPlaces Then
               strD = left(strD, intPlaces)
             End If
          End If
          ' get rid of trailing zeros if desired
          If blnTZeros = False Then
             Do
30
               blnZeroFound = False
               If right(strD, 1) = "0" Then
                  strD = left(strD, Len(strD) - 1)
                  blnZeroFound = True
               End If
35
             Loop While blnZeroFound
          End If
          ' reassemble string
```

```
If Len(strD) > 0 Then
             FormatReal = strI & "." & strD
           Else
             FormatReal = strI
 5
           End If
        End Function
        ' formats fraction as mixed number for item presentation
        Private Function FormatFraction(ByVal strFraction As String) As String
           Dim intNum As Integer
           Dim intDen As Integer
10
           Dim intQuot As Integer
          Dim vntI As Variant
           vntI = InStr(strFraction, "/")
          ' it's an integer
          If vntI = 0 Then ' it's a whole number
15,
  Ō١
             FormatFraction = strFraction
Exit Function
          End If
          intNum = CInt(left(strFraction, vntI - 1))
           intDen = CInt(right(strFraction, Len(strFraction) - vntI))
          If intDen > 0 And Abs(intNum) > intDen Then
             intQuot = Int(intNum / intDen)
             intNum = intNum Mod intDen
             FormatFraction = Trim(Str(intQuot)) & " " & Trim(Str(Abs(intNum))) & "/" & _
               Trim(Str(intDen))
25
          Else
             FormatFraction = strFraction
          End If
        End Function
        Private Function FormatUntyped(ByVal strValue As String)
30
          Dim varI As Variant
          ' see if the value is a list - if so, it will be in []
          If left(strValue, 1) = "[" And right(strValue, 1) = "]" Then
             ' trim the brackets off
```

```
FormatUntyped = Mid(strValue, 2, Len(strValue) - 2)
          Else
            FormatUntyped = strValue
          End If
 5
       End Function
       Private Function MTTextSubstitution(Source As String, dest As String)
          Dim stat
          Selection.Copy
          'Init API, reset transform
          If MTUtil.CheckMTDLLVersion = 0 Then Exit Function
10
          MTXFormReset
          'first substitution
          stat = MTXFormAddVarSub( _
            mtxfmSUBST ALL,
            mtxfmVAR_SUB_PLAIN_TEXT, Source, 0, _
151
            mtxfmVAR SUB PLAIN TEXT, dest, Len(dest), mtxfmSTYLE_NUMBER)
         If stat <> 0 Then
            MsgBox "MTXFormAddVarSub returned: " + Str(stat)
            Exit Function
         End If
          'do the substitution
         stat = TransformGraphicEquation
         If stat \Leftrightarrow 0 Then
 Ŀŝ.
            MsgBox "TransformGraphicEquation returned: " + Str(stat)
 25
            Exit Function
         End If
         MTTermAPI
          Selection.Delete
          'Paste new equation
30
         Selection.Collapse Direction:=wdCollapseEnd
          Selection.PasteSpecial Placement:=wdInLine
       End Function
```

VBSCA -380-

	' PrintModel.cls
	VERSION 1.0 CLASS
	BEGIN
_	MultiUse = -1 'True
5	END
	Attribute VB_Name = "PrintModel"
	Attribute VB_GlobalNameSpace = False
	Attribute VB_Creatable = True
10	Attribute VB_PredeclaredId = False
10	Attribute VB_Exposed = False Option Explicit
	Option Explicit
	Private mstrModelName As String
	Private mstrNow As String
•	Private mintPage As Integer
15	Private mintTab As Integer
C)	Public Property Let ModelName(ByVal strNewValue As String)
14.	
Q1	mstrModelName = strNewValue
ui F	
e.	End Property
20 mg m m m m m m m m m m m m m m m m m m	Public Sub PrintString(ByVal strS As String, ByVal intIndent As Integer)
	Tuble Sub TimeString(By Var Str5 As String, By Var intrident As integer)
25.	CheckPageBreak
LJ.	0
75.5 47.3	If Printer.CurrentY = 0 Then PrintHeading
in i	
2 5	Printer.Print Space(intIndent * mintTab) & strS
C)	
	End Sub
	Drivete Cub Drivet Leading()
	Private Sub PrintHeading()
	Dim intY As Integer
	Dim int 7 is integer
30	Printer.CurrentY = 1440 ' top margin
	Printer.Print Space(mintTab) &
	"Variables and constraints for model " & mstrModelName
	Printer.Print Space(mintTab) & mstrNow
	Printer.CurrentY = Printer.CurrentY + 100
35	Printer.Line Step(0, 0)-Step(Printer.Width, 0)
	SkipLine

```
intY = Printer.CurrentY
          Printer.CurrentY = Printer.Height - 1700
          Printer.Line Step(0, 0)-Step(Printer.Width, 0)
          Printer.CurrentY = Printer.CurrentY + 100
 5
          Printer.CurrentX = 0
          Printer.Print Space(mintTab) & "Page " & Str(mintPage)
          Printer.CurrentY = intY
          mintPage = mintPage + 1
        End Sub
10
        Private Sub SkipLine()
          Printer.Print " "
        End Sub
        Private Sub CheckPageBreak()
154
          Select Case Printer.PaperSize
            Case vbPRPSLetter, vbPRPSLetterSmall
               Call CheckOrientation(8.5, 11)
            Case vbPRPSTabloid
               Call CheckOrientation(11, 17)
            Case vbPRPSLedger
               Call CheckOrientation(17, 11)
            Case vbPRPSLegal
               Call CheckOrientation(8.5, 14)
 U)
 C)
          End Select
 þá
25]
        End Sub
 Private Sub CheckOrientation(ByVal sngWidth As Single,
          ByVal sngHeight As Single)
          'convert inches to twips
30
          sngWidth = sngWidth * 1440
          sngHeight = sngHeight * 1440
          If Printer.Orientation = vbPRORPortrait Then
            If Printer.CurrentY >= sngHeight - 2200 Then
               Printer.NewPage
35
            End If
          Else
            If Printer.CurrentY >= sngWidth - 2200 Then
```

```
Printer.NewPage
           End If
         End If
 5
       End Sub
       Private Sub Class_Initialize()
         Printer.FontSize = 11
         mstrNow = Now
         mintPage = 1
10
         mintTab = 4
       End Sub
       Private Sub Class_Terminate()
         Printer.EndDoc
End Sub
```

```
' Progress.cls
        VERSION 1.0 CLASS
        BEGIN
         MultiUse = -1 'True
 5
        END
        Attribute VB Name = "Progress"
        Attribute VB GlobalNameSpace = False
        Attribute VB Creatable = True
        Attribute VB PredeclaredId = False
10
        Attribute VB Exposed = False
        ' class to give visual indication of progress
        Option Explicit
        Private mintStepSize As Integer
        ' pulls up form
       Public Sub Init(ByVal intNumIncrements As Integer, _
15
          Optional ByVal strCaption As String)
  ųĵ
  Ø1
          If intNumIncrements = 0 Then 'prevent divide by 0
            Beep
            Exit Sub
          End If
          mintStepSize = 500 / intNumIncrements
          frmProgress.prbProgressBar.Max = mintStepSize * intNumIncrements
          If Len(strCaption) > 0 Then
            frmProgress.lblProgress = strCaption
.
C)
          End If
30
          frmProgress.Show
          frmProgress.Refresh
       End Sub
       'bumps the progress bar to the next increment. When the progress
        'bar is fully advanced, the form is unloaded.
35
       Public Sub Advance()
          Dim intStop As Integer
          With frmProgress.prbProgressBar
            If .Value = .Max Then
40
```

```
Exit Sub
           End If
           intStop = .Value + mintStepSize
           Do Until .Value = intStop
              .Value = .Value + 1
 5
              If .Value = .Max Then
                Unload frmProgress
                Exit Sub
              End If
10
            Loop
         End With
       End Sub
       Public Sub AbsoluteAdvance(ByVal intNewValue As Integer)
15
         frmProgress.prbProgressBar.Value = intNewValue * mintStepSize
       End Sub
   C)
  Public Sub Kill()
         Unload frmProgress
       End Sub
```

	' Prolog.cls VERSION 1.0 CLASS
	BEGIN
	MultiUse = 0 'False
5	Persistable = 0 'NotPersistable
	DataBindingBehavior = 0 'vbNone
	DataSourceBehavior = 0 'vbNone
	MTSTransactionMode = 0 'NotAnMTSObject
	END
10	Attribute VB_Name = "Prolog"
	Attribute VB_GlobalNameSpace = False
	Attribute VB_Creatable = True
	Attribute VB_PredeclaredId = False
	Attribute VB_Exposed = True
15	Attribute VB_Ext_KEY = "SavedWithClassBuilder", "Yes"
	Attribute VB_Ext_KEY = "Top_Level", "Yes"
	Option Explicit
2	Private Declare Function StartProlog4Session Lib "prlghlapi.dll" _
## #4	(ByVal strP4FN As String) As Long
20 ₁₁	Private Declare Function EndProlog4Session Lib "prlghlapi.dll" () As Long
er er	Private Declare Function GetHLAPIVersion Lib "prlghlapi.dll" () As String
41	Private Declare Function VBGetHLAPIVersion Lib "prlghlapi.dll" () As String
nýu Eu	Private Declare Function SolveConstraintOrdered Lib "prlghlapi.dll" _
i.	(ByVal Constraint As String, ByVal SolutionOrder As Long) As Long
25	Private Declare Function SolveConstraintRandomly Lib "prlghlapi.dll" _
	(ByVal Constraint As String) As Long
	Private Declare Function SolveConstraintOrderedNSolns Lib "prlghlapi.dll" _
	(ByVal Constraint As String, ByVal SolutionOrder As Long, _
19 6	ByVal NumSols As Long) As Long
30	Private Declare Function IsFullyConstrained Lib "prlghlapi.dll" _
<u> </u>	(ByVal Constraint As String) As Long
	Private Declare Function GetValue Lib "prlghlapi.dll"
	(ByVal strVarName As String) As Long
	Private Declare Function VBGetValue_string Lib "prlghlapi.dll" _
35	(ByVal udtPtr As Any) As String
	Private Declare Function VBPrintAllVarVals Lib "prlghlapi.dll" () As String
	Private Declare Function SetSolnDiffWt Lib "prlghlapi.dll" _
	(ByVal Weight As Long) As Long
	Private Declare Function SetPrologInterruptFile Lib "prlghlapi.dll" _
40	(ByVal strFN As String) As Long
	'Keep the constants in sync with appropriate values in prlghlapi.h
	'Solution-Orders:

```
Private Enum PrologOrder
          prDontCareOrder = 0
          prDifferentOrder = 10
          prLikeOrder = 20
 5
          prRandomOrder = 30
          prUniqueOrder = 40
        End Enum
        Private Enum PrologType
          prValUnknown = 0
          prValInteger = 10
10
          prValRationalFloat = 12
          prValRationalFraction = 13
          prValIrrational = 14
          prValReal = 15
          prValString = 20
15
          prValList = 25
          prValFunctor = 30
          prValSymbol = 35
          prValVar = 100
        End Enum
 Pen albu Hin alon Tim
        Private Enum PrologErrors
          prErrInitialization = -10
          prErrIntegerraintTooLong = -15
          prErrGettingTerm = -20
prErrMakingFunctor = -25
          prErrInvalidInterval = -30
          prErrArityTooMany = -35
          prErrParse = -40
          prErrNullTerm = -45
        End Enum
        ' used to hold all strings for the Prolog
        Private mcolVNs As Collection
        Private mstrDelimit As String
        Private mintNumSols As Integer
35
        Event Finished(ByVal lngRet As Long)
        Private Sub Class Initialize()
          Set mcolVNs = New Collection
```

Set gProlog = Me 'gProlog is defined in Timer.bas Dim lngRet As Long 5 ' if this file exists, interrupt prolog processing lngRet = SetPrologInterruptFile("c:\halt.tca") End Sub Private Sub Class_Terminate() 10 Set gProlog = Nothing End Sub Public Property Get Version() As String Version = GetHLAPIVersion() **End Property** ' sets the degree of difference in the variants. Range is 0 to 2. Public Property Let DiffWeight(ByVal bytDifference As Byte) Call SetSolnDiffWt(CLng(bytDifference)) **End Property** Public Function StartProlog() As Boolean ChDir App. Path ' set path to application dir for hlp4lib.p4 file StartProlog = CBool(StartProlog4Session("hlp4lib.p4")) **End Function** Public Function EndProlog() As Boolean ChDir App. Path ' set path to application dir for hlp4lib.p4 file EndProlog = CBool(EndProlog4Session()) 30 **End Function** Public Sub AddVariable(ByVal strS As String)

```
If Len(strS) > 0 Then 'it's not an untyped variable
            Call mcolVNs.Add(strS)
            mstrDelimit = "end var defs,"
          End If
 5
       End Sub
       Public Sub AddConstraint(ByVal strS As String)
          Call mcolVNs.Add(mstrDelimit & strS)
          mstrDelimit = ""
        End Sub
10
       Public Sub SolveConstraintsRandomly()
          SolveAsync ' in Timer.bas - must be in a standard module
       End Sub
       Public Sub SolveConstraintsAsync()
Dim strS As String
          Dim lngRet As Long
          lngRet = -1 ' default to error condition
          If mcolVNs.Count > 0 Then 'there's something for Prolog to chew on
            strS = BuildString()
            ChDir App.Path ' set path to application dir for hlp4lib.p4 file
            lngRet = SolveConstraintRandomly(strS) ' call Prolog
          End If
          RaiseEvent Finished(lngRet)
          Set mcolVNs = New Collection
30
       End Sub
       Private Function RandomNumSols() As Integer
          Randomize
          RandomNumSols = 10 * Rnd - 0.5
          If RandomNumSols = 0 Then RandomNumSols = 1
35
```

Private Sub Advance(ByVal IngRet As Long) Dim intI As Integer 5 For intI = 1 To lngRetNextSolution Next intI **End Sub** ' gets the next solution, returns true if one exists, false if it doesn't 10 Private Function NextSolution() As Boolean ChDir App.Path 'set path to application dir for hlp4lib.p4 file NextSolution = SolveConstraintOrderedNSolns(vbNullString, _ prUniqueOrder, mintNumSols) **End Function** Public Property Get PrintAllVals() As String M. H. L. PrintAllVals = VBPrintAllVarVals 20 **End Property** ' get the values associated with each solution Public Property Get Value(ByVal strVN As String) As String Dim lngPtr As Long Dim strT As String ChDir App.Path ' set path to application dir for hlp4lib.p4 file lngPtr = GetValue(strVN) ' returns a pointer to the variable If lngPtr Then ' to handle untyped variables that have no constraint, and therefore no value 30 strT = VBGetValue string(lngPtr) ' returns a string Value = Left(strT, Len(strT) - 1) ' trim off the null delimiter Else Value = " " End If 35

End Function

End Property

Private Function BuildString() As String

```
Dim varStr As Variant
          Dim strS As String
          For Each varStr In mcolVNs
             strS = strS & varStr & ", "
 5
          Next varStr
          ' trim off the last comma and space
          strS = Left(strS, Len(strS) - 2)
          ' add a period
10
          strS = strS \& "."
          BuildString = strS
        End Function
15
        Public Sub ShowString()
 ij,
```

Dim strS As String

strS = BuildString()
' Call MsgBox(strS, , "Prolog string is:")

End Sub

	' PSMODEL.cls
	VERSION 1.0 CLASS
	BEGIN
_	MultiUse = -1 'True
5	Persistable = 0 'NotPersistable
	DataBindingBehavior = 0 'vbNone
	DataSourceBehavior = 0 'vbNone
	MTSTransactionMode = 0 'NotAnMTSObject
	END
10	Attribute VB_Name = "SMCModel"
	Attribute VB_GlobalNameSpace = False
	Attribute VB_Creatable = False
	Attribute VB_PredeclaredId = False
	Attribute VB_Exposed = False
15	Option Explicit
	Implements Model
٣٩	
e. Mi	Dim mudtModel As Model
	Dim lastStart As Integer
U1	
## ##	Private Sub Class_Initialize()
Ä.	
ար ար ար և այլան և և արե և և այլ կար այլ հայի հայի հայի հայի 20	Set mudtModel = New Model
e F	End Sub
la # Lij	
	'Delegated to Class Model
i i	Public Property Get Model_FileName() As String
Hill Rule Him Hills Him, Hills Huns Huns Huns Huns Huns Huns Huns Hun	N. 11 PH N. 14 PH N
	Model_FileName = mudtModel.FileName
25	End Property
	' Delegated to Class Model
	Public Property Let Model_FileName(ByVal strNewValue As String)
	mudtModel.FileName = strNewValue
	End Property
30	' Delegated to Class Model
	Public Property Get Model IsFrozen() As Boolean

	Model_IsFrozen = mudtModel.IsFrozen
	End Property
	'Delegated to Class Model Public Property Let Model_IsFrozen(ByVal blnNewValue As Boolean)
5	mudtModel.IsFrozen = blnNewValue
	End Property
	'Delegated to Class Model Public Sub Model_AddChecksum(ByVal dblChecksum As Double)
	Call mudtModel.AddChecksum(dblChecksum)
10	End Sub
	' Delegated to Class Model Public Sub Model_InitChecksums()
C.	mudtModel.InitChecksums
գեր բեզ ընտ կետ կեր ինել Բ'' Դր հեմ Դ և տա ահ հետ հահ հետ հ	End Sub
151	'Delegated to Class Model Public Sub Model_InitTempChecksums()
	mudtModel.InitTempChecksums
	End Sub
20	' Delegated to Class Model Public Function Model_ChecksumExists(ByVal dblChecksum As Double) As Boolean
	Model_ChecksumExists = mudtModel.ChecksumExists(dblChecksum)
	End Function
	' Delegated to Class Model Public Property Get Model_Comments() As String
25	Model_Comments = mudtModel.Comments
	End Property

' Delegated to Class Model Public Property Let Model Comments(ByVal strNewValue As String) mudtModel.Comments = strNewValue **End Property** ' Delegated to Class Model Public Property Get Model Clones() As CClones Set Model Clones = mudtModel.Clones **End Property** ' Delegated to Class Model Public Property Get Model Variables() As CVariables Set Model_Variables = mudtModel.Variables **End Property** ' Delegated to Class Model Public Property Get Model Constraints() As CConstraints Set Model Constraints = mudtModel.Constraints **End Property** ' Delegated to Class Model Public Property Let Model_IsDirty(ByVal blnNewValue As Boolean) mudtModel.IsDirty = blnNewValue **End Property** ' Delegated to Class Model Public Property Get Model IsDirty() As Boolean Model IsDirty = mudtModel.IsDirty **End Property** ' Delegated to Class Model Public Property Let Model LastClone(ByVal intNewValue As Integer)

5

10

4J

15

17 1ft. f

20

25

mudtModel.LastClone = intNewValue

End Proper	rtv
------------	-----

' Delegated to Class Model Public Property Get Model_LastClone() As Integer

5 Model LastClone = mudtModel.LastClone

End Property

' Delegated to Class Model Public Sub Model FreezeModel()

Call mudtModel.FreezeModel

10 End Sub

J)

15

20

' Delegated to Class Model Public Sub Model OpenDoc(ByVal udtWord As MSWord)

Call mudtModel.OpenDoc(udtWord)

End Sub

'Delegated to Class Model Public Sub Model_CloseDoc()

Call mudtModel.CloseDoc

End Sub

'Delegated to Class Model Public Sub Model CloseAllCloneDocs()

Call mudtModel.CloseAllCloneDocs

End Sub

' Delegated to Class Model Public Sub Model_ReadModel()

25 mudtModel.ReadModel

End Sub

VBSCA -395-

' Delegated to Class Model Public Sub Model ReadObjects() mudtModel.ReadObjects End Sub ' Delegated to Class Model Public Sub Model WriteModel() mudtModel.WriteModel End Sub ' Delegated to Class Model Public Sub Model WriteObjects() 10 mudtModel.WriteObjects End Sub IJ, ' Delegated to Class Model Public Function Model ConstraintsOK(ByVal udtTestType As TestType, ByVal udtProlog As Prolog, blnUnderconstrained As Boolean, blnTestAborted As Boolean, strUnderconstrainedVN As String) As Boolean Model ConstraintsOK = mudtModel.ConstraintsOK(udtTestType, udtProlog, _ blnUnderconstrained, blnTestAborted, strUnderconstrainedVN) **End Function** ' implemented here Public Sub Model GenerateClones(ByVal udtWord As MSWord, ByVal udtProlog As Prolog, ByVal intNumClones As Integer, ByVal bytDifference As Byte) Call mudtModel.SubstituteValues(Me, udtWord, udtProlog, intNumClones, _ bytDifference, 50) End Sub ' Delegated to Class Model Public Sub Model SubstituteValues(ByVal objO As Object, ByVal udtWord As MSWord, ByVal udtProlog As Prolog, ByVal intNumClones As Integer, ByVal bytDifference As Byte, _ ByVal intStartPos As Integer) 30

End Sub

```
Public Sub CreateVariant(ByVal udtClone As Clone)
          With udtClone.CloneDoc.Bookmarks
           If .Exists("tca RespA") = False Or
             .Exists("tca RespB") = False Or
             .Exists("tca RespC") = False Or _
             .Exists("tca RespD") = False Or
             .Exists("tca RespE") = False Or
             .Exists("tca Key") = False Then
             Call MsgBox("Model is missing a TCA Bookmark!", vbExclamation, "Hey!")
10
             Exit Sub
           End If
          End With
          Dim nchoices As Integer
          Dim lowerbound As Integer
          Dim upperbound As Integer
          nchoices = 5
          lowerbound = 1
          upperbound = 8
201
         Dim resp(10) As String
         Dim used(10) As Integer
         resp(0) = udtClone.CloneDoc.Bookmarks("key").Range.Text
         Dim i As Integer
         For i = lowerbound To upperbound
           used(i) = 0
           resp(i) = udtClone.CloneDoc.Bookmarks("resp" & Format(i)).Range.Text
         Next
         Dim nselected As Integer
         nselected = 0
30
         Dim rnumber As Integer
         Dim rnumbers(10) As Integer
          While (nselected < upperbound)
```

rnumber = (upperbound - lowerbound + 1) * Rnd + lowerbound - 0.5

If (rnumber > upperbound) Then

```
rnumber = upperbound
             End If
             If (used(rnumber) = 0) Then
               used(rnumber) = 1
               nselected = nselected + 1
 5
               rnumbers(nselected) = rnumber
             End If
           Wend
           Dim unsorted(10) As Integer
10
           unsorted(0) = 0
           nselected = 0
           Dim j As Integer
           Dim n As Integer
           Dim crStr As String
           Dim tabcrStr As String
           crStr = Chr(13)
           tabcrStr = Chr(9) \& Chr(13)
           For i = lowerbound To upperbound
             If resp(rnumbers(i)) \Leftrightarrow tabcrStr And
               resp(rnumbers(i)) \Leftrightarrow crStr And
               Mid(resp(rnumbers(i)), 1, 10) <> "Distractor" Then
               n = 0
               For j = 0 To nselected
                 If IsNumeric(resp(rnumbers(i))) = True And
                   IsNumeric(resp(unsorted(j))) = True And
                   Asc(resp(rnumbers(i))) \Leftrightarrow 36 Then ' 36 is the $ sign
                   If Val(resp(rnumbers(i))) = Val(resp(unsorted(j))) Then
                     If Asc(resp(rnumbers(i))) \Leftrightarrow 1 Then
                        n = 1
                        Exit For
30
                     End If
                   End If
                 Else
                   If resp(rnumbers(i)) = resp(unsorted(j)) Then
35
                     If Asc(resp(rnumbers(i))) \Leftrightarrow 1 Then
                        n = 1
                        Exit For
                     End If
                   End If
```

```
End If
              Next
              If n = 0 Then
                nselected = nselected + 1
 5
                unsorted(nselected) = rnumbers(i)
                If nselected = nchoices - 1 Then
                If nselected = upperbound Then
                  Exit For
                End If
              End If
10
            End If
           Next
           For i = 0 To nselected
             used(i) = 0
15
          Next
          Dim sorted(10) As Integer
          Dim resp1, resp2 As String
 ű)
          Dim val1, val2 As Variant
 đ1
          For i = 0 To nselected
            For j = 0 To neelected
              If (used(i) = 0) Then
                sorted(i) = unsorted(j)
               n = j
                Exit For
              End If
            Next
            For j = 0 To nselected
              If (used(i) = 0) Then
                resp1 = resp(unsorted(j))
30
                resp2 = resp(sorted(i))
                If left(resp1, 1) = "\$" Then
                  val1 = Val(right(resp1, Len(resp1) - 1))
                Else
                  val1 = Val(resp1)
35
                End If
                If left(resp2, 1) = "\$" Then
                  val2 = Val(right(resp2, Len(resp2) - 1))
```

```
Else
                  val2 = Val(resp2)
                End If
                If (val1 < val2) Then
 5
                  sorted(i) = unsorted(j)
                End If
              End If
            Next
            used(n) = 1
10
          Next
          For i = 0 To nselected
            If sorted(i) = 0 Then
              Exit For
            End If
          Next
          Dim min, max As Integer
          min = i - 4
          If min < 0 Then
201
            min = 0
          End If
          max = i
          If max > nselected - 4 Then
            max = nselected - 4
          End If
          If max < 0 Then
            max = 0
          End If
          Dim iStart As Integer
30
          Dim iEnd As Integer
          If max > 0 And max + 4 \le nselected Then
            iStart = lastStart
            While iStart = lastStart
              iStart = (max - min + 1) * Rnd + min - 0.5
35
            Wend
```

```
lastStart = iStart
            iEnd = iStart + nchoices - 1
          Else
            iStart = 0
            If nselected > 4 Then
 5
              iEnd = 4
            Else
              iEnd = nselected
            End If
10
            lastStart = iStart
          End If
          Dim respRange As Range
          Dim choice As String
          Dim key As String
15
          n = 1
          For i = iStart To iEnd
            choice = Mid("ABCDE", n, 1)
            If sorted(i) = 0 Then
              udtClone.CloneDoc.Bookmarks("key").Range.Copy
            Else
              udtClone.CloneDoc.Bookmarks("resp" & Format(sorted(i))).Range.Copy
            End If
            Set respRange = udtClone.CloneDoc.Bookmarks("tca Resp" & choice).Range
            respRange.Paste
            respRange.Borders.Enable = False
            respRange.Borders.InsideLineStyle = wdLineStyleNone
            udtClone.CloneDoc.Bookmarks.Add name:="tca Resp" & choice, Range:=respRange
            respRange.InsertBefore Text:=choice & ". "
            If sorted(i) = 0 Then
30
             key = choice
              udtClone.key = choice
            End If
            n = n + 1
          Next
35
          For i = nselected + 1 To nchoices - 1
```

choice = Mid("ABCDE", i + 1, 1)
Set respRange = udtClone.CloneDoc.Bookmarks("tca_Resp" & choice).Range
respRange.Text = "[NO VALUE]" & Chr(13) & Chr(10)
udtClone.CloneDoc.Bookmarks.Add name:="tca_Resp" & choice, Range:=respRange
respRange.InsertBefore Text:=choice & ". "
Next

Dim keyRange As Range Set keyRange = udtClone.CloneDoc.Bookmarks("tca_Key").Range keyRange.InsertBefore Text:="Key is " & key

10 End Sub

5

	' QCModel.cls
	VERSION 1.0 CLASS
	BEGIN
_	MultiUse = -1 'True
5	Persistable = 0 'NotPersistable
	DataBindingBehavior = 0 'vbNone DataSourceBehavior = 0 'vbNone
	MTSTransactionMode = 0 'NotAnMTSObject
	END
10	Attribute VB Name = "QCModel"
10	Attribute VB GlobalNameSpace = False
	Attribute VB_Global values page 1 also Attribute VB Creatable = False
	Attribute VB PredeclaredId = False
	Attribute VB_Exposed = False
15	Option Explicit
	•
	Implements Model
£	
11	Dim mudtModel As Model
स्तिता होष्टर बीचा भेडाल प्रदेश होष्टर 179 उन्हें परस्के मून्न स्थान स्थान होता है	Drivete Sub Class Initializad
111	Private Sub Class_Initialize()
34	Set mudtModel = New Model
wa It	Set mudiviouer - New Moder
204	End Sub
E	
C)	' Delegated to Class Model
4 <u>1</u>	Public Property Get Model_FileName() As String
E 54 E E E 65	
	Model_FileName = mudtModel.FileName
	T. I.D.
	End Property
25	' Delegated to Class Model
23	Public Property Let Model_FileName(ByVal strNewValue As String)
	Tuble Troporty Est Wodel_Thertaine(E) various value Tie Samigy
	mudtModel.FileName = strNewValue
	End Property
20	'Delegated to Class Model
30	Public Property Get Model_IsFrozen() As Boolean
	Model IsFrozen = mudtModel.IsFrozen
	1410de1_131 102e11 - Illudui410de1.131 102e11

	End Property
	' Delegated to Class Model Public Property Let Model_IsFrozen(ByVal blnNewValue As Boolean)
	mudtModel.IsFrozen = blnNewValue
5	End Property
	' Delegated to Class Model Public Property Get Model_Comments() As String
	Model_Comments = mudtModel.Comments
	End Property
10	'Delegated to Class Model Public Property Let Model_Comments(ByVal strNewValue As String)
	mudtModel.Comments = strNewValue
	End Property
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1	' Delegated to Class Model Public Property Get Model_Clones() As CClones
an Ferral	Set Model_Clones = mudtModel.Clones
	End Property
The part of the second	'Delegated to Class Model Public Property Get Model_Variables() As CVariables
20	Set Model_Variables = mudtModel.Variables
	End Property
	' Delegated to Class Model Public Property Get Model_Constraints() As CConstraints
	Set Model_Constraints = mudtModel.Constraints
25	End Property
	'Delegated to Class Model

Public Sub Model AddChecksum(ByVal dblChecksum As Double) Call mudtModel.AddChecksum(dblChecksum) End Sub ' Delegated to Class Model Public Sub Model InitChecksums() 5 mudtModel.InitChecksums End Sub ' Delegated to Class Model Public Sub Model InitTempChecksums() mudtModel.InitTempChecksums 10 End Sub 'Delegated to Class Model Public Function Model ChecksumExists(ByVal dblChecksum As Double) As Boolean Model ChecksumExists = mudtModel.ChecksumExists(dblChecksum) **End Function** ' Delegated to Class Model Public Property Let Model IsDirty(ByVal blnNewValue As Boolean) mudtModel.IsDirty = blnNewValue**End Property** 20 ' Delegated to Class Model Public Property Get Model IsDirty() As Boolean Model IsDirty = mudtModel.IsDirty **End Property** ' Delegated to Class Model Public Property Let Model LastClone(ByVal intNewValue As Integer) 25 mudtModel.LastClone = intNewValue

End Property

' Delegated to Class Model Public Sub Model_FreezeModel()

Call mudtModel.FreezeModel

- 5 End Sub
 - ' Delegated to Class Model Public Property Get Model LastClone() As Integer

Model LastClone = mudtModel.LastClone

End Property

'Delegated to Class Model Public Sub Model_OpenDoc(ByVal udtWord As MSWord)

Call mudtModel.OpenDoc(udtWord)

End Sub

C)

43 61

M. Han Can

15

'Delegated to Class Model Public Sub Model_CloseDoc()

Call mudtModel.CloseDoc

End Sub

'Delegated to Class Model Public Sub Model CloseAllCloneDocs()

20 Call mudtModel.CloseAllCloneDocs

End Sub

'Delegated to Class Model Public Sub Model_ReadModel()

mudtModel.ReadModel

- 25 End Sub
 - ' Delegated to Class Model

VBSCA -406-

Public Sub Model ReadObjects() mudtModel.ReadObjects End Sub ' Delegated to Class Model Public Sub Model_WriteModel() mudtModel.WriteModel End Sub ' Delegated to Class Model Public Sub Model_WriteObjects() mudtModel.WriteObjects 10 **End Sub** 41 ' Delegated to Class Model Public Function Model ConstraintsOK(ByVal udtTestType As TestType, __ ByVal udtProlog As Prolog, blnUnderconstrained As Boolean, _ blnTestAborted As Boolean, strUnderconstrainedVN As String) As Boolean Model ConstraintsOK = mudtModel.ConstraintsOK(udtTestType, udtProlog, blnUnderconstrained, blnTestAborted, strUnderconstrainedVN) **End Function** 'implemented here Public Sub Model GenerateClones(ByVal udtWord As MSWord, ByVal udtProlog As Prolog, _ ByVal intNumClones As Integer, ByVal bytDifference As Byte) Call mudtModel.SubstituteValues(Me, udtWord, udtProlog, intNumClones, bytDifference, 275) End Sub ' Delegated to Class Model 25 Public Sub Model SubstituteValues(ByVal objO As Object, _ ByVal udtWord As MSWord, ByVal udtProlog As Prolog, ByVal intNumClones As Integer, ByVal bytDifference As Byte, _ ByVal intStartPos As Integer) 30

End Sub

Dim columnAValue Dim columnBValue

```
Public Sub CreateVariant(ByVal udtClone As Clone)
          Dim rnumber As Integer
          Dim sLen As Integer
 5
          Dim columnRange As Range
          Dim columnAValStr As String
          Dim columnBValStr As String
          With udtClone.CloneDoc
            rnumber = .Tables(2).Rows.Count * Rnd + 0.5
10
            .Tables(2).Cell(Row:=rnumber, Column:=1).Range.Copy
            columnAValStr = .Tables(2).Cell(Row:=rnumber, Column:=2).Range.Text
            sLen = Len(columnAValStr)
            If sLen > 1 Then
              columnAValStr = left(columnAValStr, sLen - 1)
            End If
            Set columnRange = .Bookmarks("tca ColumnA").Range
            columnRange.Paste
            rnumber = .Tables(3).Rows.Count * Rnd + 0.5
 ij
            .Tables(3).Cell(Row:=rnumber, Column:=1).Range.Copy
            columnBValStr = .Tables(3).Cell(Row:=rnumber, Column:=2).Range.Text
20≇
            sLen = Len(columnBValStr)
            If sLen > 1 Then
              columnBValStr = left(columnBValStr, sLen - 1)
            End If
            Set columnRange = .Bookmarks("tca ColumnB").Range
25
            columnRange.Paste
            If .Tables(1).Columns.Count = 4 Then ' fixes weird behavior if only 1 row in model
              .Tables(1).Cell(Row:=1, Column:=4).Delete
              .Tables(1).Cell(Row:=1, Column:=3).Delete
            End If
30
            Dim key As String
```

```
If IsNumeric(columnAValStr) = True And
             IsNumeric(columnBValStr) = True Then
             columnAValue = Val(columnAValStr)
             columnBValue = Val(columnBValStr)
             If columnAValue > columnBValue Then
 5
               key = "A"
             ElseIf columnBValue > columnAValue Then
               key = "B"
             ElseIf columnAValue = columnBValue Then
               key = "C"
10
             End If
           End If
         End With
         Dim keyRange As Range
         Set keyRange = udtClone.CloneDoc.Bookmarks("tca_Key").Range
         If key = "" Then
           keyRange.InsertBefore Text:="TCA cannot determine the key"
           keyRange.InsertBefore Text:="Key is " & key
         End If
 udtClone.key = key
       End Sub
 C)
```

```
'StringSolver.cls
       VERSION 1.0 CLASS
       BEGIN
        MultiUse = 0 'False
        Persistable = 0 'NotPersistable
 5
        DataBindingBehavior = 0 'vbNone
        DataSourceBehavior = 0 'vbNone
        MTSTransactionMode = 0 'NotAnMTSObject
       END
       Attribute VB Name = "StringSolver"
10
       Attribute VB GlobalNameSpace = False
       Attribute VB Creatable = True
       Attribute VB PredeclaredId = False
       Attribute VB Exposed = False
15
       Option Explicit
       Dim mudtVS As VarString
       Dim mcolValues As Collection
 ű.
       Public Property Let StringVariable(ByVal udtNewValue As VarString)
         Set mudtVS = udtNewValue
       End Property
       Public Property Get RandomValueCollection() As Collection
         Dim udtSS As SubString
         Dim strS As String
         Dim varS As Variant
         Set mcolValues = New Collection
         strS = mudtVS.StringCollection.Item(GetRandomIndex)
30
         If mudtVS.IsIndexed Then
            Set udtSS = New SubString
            udtSS.Delimiter = mudtVS.Delimiter
            udtSS.StringValue = strS
35
            For Each varS In udtSS.StringCollection
              Call mcolValues.Add(varS)
            Next varS
         Else
```

Call mcolValues.Add(strS)

End If

Set RandomValueCollection = mcolValues

5 End Property

Private Function GetRandomIndex() As Integer

Dim intl As Integer

intI = mudtVS.StringCollection.Count * Rnd + 0.5

'Seems to produce an out-of-range value sometimes.

'This will fix it.

If intI < 1 Then intI = 1

If intI > mudtVS.StringCollection.Count Then intI = mudtVS.StringCollection.Count

GetRandomIndex = intI

End Function

'StringSolverx.cls
VERSION 1.0 CLASS
BEGIN
MultiUse = -1 'True

5 END
Attribute VB_Name = "StringSolver"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False

10 Attribute VB_Exposed = False
Option Explicit

Private mcolSV As Collection

Private Sub Class_Initialize()

Set mcolSV = New Collection

End Sub

```
'SubString.cls
        VERSION 1.0 CLASS
       BEGIN
         MultiUse = -1 'True
 5
       END
        Attribute VB Name = "SubString"
       Attribute VB GlobalNameSpace = False
        Attribute VB Creatable = True
       Attribute VB_PredeclaredId = False
       Attribute VB Exposed = False
10
        Option Explicit
        Private mstrDelimiter As String
       Private mstrString As String
       Private mcolStr As Collection
Private Sub Class Initialize()
          Set mcolStr = New Collection
       End Sub
       Public Property Let Delimiter(ByVal strNewValue As String)
20-
          mstrDelimiter = strNewValue
 ű
       End Property
        ' use this to convert a concatenated string to a collection
       Public Property Let StringValue(ByVal strNewValue As String)
25
          mstrString = strNewValue
       End Property
       ' or use this to convert a collection to a concatenated string
       Public Property Let StringCollection(ByVal colNewValue As Collection)
30
          Set mcolStr = colNewValue
       End Property
```

	Public Property Get StringValue() As String
5	Dim varS As Variant Dim strS As String
	'build new string For Each varS In mcolStr strS = strS & varS & mstrDelimiter Next varS
10	' trim last character If Len(strS) > 0 Then StringValue = left(strS, Len(strS) - 1) End If
15	End Property
	' converts concatenated string into a collection Public Property Get StringCollection() As Collection
0 0 20 4 4 4 3	Dim colC As New Collection Dim intI As Integer
8	For intI = 1 To NumSubStrings Call colC.Add(GetSubString(intI)) Next intI
251 11 11 14 15	Set StringCollection = colC
# . .	End Property
5) 30	' returns the number of substrings in this string Public Property Get NumSubStrings() As Integer
	Dim intD As Integer Dim intI As Integer Dim varS As Variant
35	If Len(mstrString) = 0 Then NumSubStrings = 0 Exit Property End If
40	For intI = 1 To Len(mstrString)

```
If Mid(mstrString, intI, 1) = mstrDelimiter Then
               intD = intD + 1
             End If
          Next intI
 5
          NumSubStrings = intD + 1
        End Property
        Public Sub AddSubString(ByVal strNewValue As String)
10
          Call mcolStr.Add(strNewValue)
        End Sub
        ' parses the substring from the string depending on intIndex
        Public Function GetSubString(ByVal intIndex As Integer) As String
          ' see if index is valid for the current string
          If NumSubStrings < intIndex Then
             GetSubString = ""
             Exit Function
          End If
          'index into the string using delimiter
          Dim varI1 As Variant
          Dim varI2 As Variant
 Dim intCount As Integer
25🖺
          varI2 = 0
          Do
             varI1 = varI2
30
             varI2 = InStr(varI1 + 1, mstrString, mstrDelimiter)
             intCount = intCount + 1
             If varI2 = 0 Then
               varI2 = Len(mstrString) + 1
             End If
35
          Loop Until intCount = intIndex
          GetSubString = Mid(mstrString, varI1 + 1, varI2 - varI1 - 1)
```

End Function

VBSCA -416-

	'Value.cls
	VERSION 1.0 CLASS
	BEGIN
_	MultiUse = -1 'True
5	END
	Attribute VB_Name = "Value"
	Attribute VB_GlobalNameSpace = False
	Attribute VB_Creatable = True
1.0	Attribute VB_PredeclaredId = False
10	Attribute VB_Exposed = False
	Option Explicit
	Dim mstrVariableName As String
	Dim mstrValue As String
	Dim mblnChecksum As Boolean
15	Dim mstrPrologString As String
	Dim mudtVariableType As VariableType
,==a	
19.4 Fig.	Public Property Get VariableName() As String
<u>a</u>	VI - I - I I - NI
	VariableName = mstrVariableName
20=	End Property
- 42	End Property
## # ## #	Public Property Let VariableName(ByVal strNewValue As String)
B.	
	mstrVariableName = strNewValue
4]	
L.	End Property
E)	Dublic Droporty Cat Value() As String
	Public Property Get Value() As String
25	Value = mstrValue
	End Property
	Public Property Let Value(ByVal strNewValue As String)
	mstrValue = strNewValue
30	End Property
50	Lita I Topolity
	Public Property Get Checksum() As Boolean

Checksum = mblnChecksum

End Property

	'VarFraction.cls
	VERSION 1.0 CLASS
	BEGIN
	MultiUse = -1 'True
5	END
_	Attribute VB Name = "VarFraction"
	Attribute VB GlobalNameSpace = False
	Attribute VB Creatable = True
	Attribute VB_PredeclaredId = False
10	Attribute VB_Exposed = False
10	Option Explicit
	Option Explicit
	Implements Variable
	Private mudtVar As Variable
	' current version of data produced by this class
15_	Const mintVERSIONSTAMP As Integer = 1
4]	Private mstrFromNum As String
₩1 139	Private mstrFromDen As String
UI F	Private mstrToNum As String
	Private mstrToDen As String
20	Private mstrByNum As String
ui	Private mstrByDen As String
E	Private mblnMixedNumbers As Boolean
C)	Private mblnIsIndependent As Boolean
15	•
	Private Sub Class_Initialize()
ļak As	
25 <u>-</u>	Set mudtVar = New Variable
μļ	
	End Sub
	Private Sub Class_Terminate()
	Set mudtVar = Nothing
30	
	End Sub
	Delegated to Class Variable
	Public Property Get Variable_Name() As String
25	Manighta Nama - may 44May Nama
35	Variable_Name = mudtVar.Name

```
End Property
      'Delegated to Class Variable
      Public Property Let Variable Name(ByVal RHS As String)
        mudtVar.Name = RHS
      End Property
      'Delegated to Class Variable
      Public Property Let Variable Typ(ByVal udtNewValue As VariableType)
        mudtVar.Typ = udtNewValue
      End Property
      ' Delegated to Class Variable
      Public Property Get Variable Typ() As VariableType
T.
        Variable_Typ = mudtVar.Typ
      End Property
      ' Delegated to Class Variable
      Public Property Get Variable Index() As Long
        Variable Index = mudtVar.Index
      End Property
      'Delegated to Class Variable
      Public Property Let Variable_Index(ByVal lngNewValue As Long)
        mudtVar.Index = lngNewValue
     End Property
      'Delegated to Class Variable
      Public Property Get Variable Enabled() As Boolean
        Variable Enabled = mudtVar.Enabled
      End Property
      'Delegated to Class Variable
```

5

10

25

```
Public Property Let Variable Enabled(ByVal RHS As Boolean)
  mudtVar.Enabled = RHS
End Property
' Delegated to Class Variable
Public Property Get Variable IsDirty() As Boolean
  Variable IsDirty = mudtVar.IsDirty
End Property
'Delegated to Class Variable
Public Property Let Variable IsDirty(ByVal RHS As Boolean)
  mudtVar.IsDirty = RHS
End Property
' Delegated to Class Variable
Public Property Get Variable_Checksum() As Boolean
  Variable_Checksum = mudtVar.Checksum
End Property
' Delegated to Class Variable
Public Property Let Variable Checksum(ByVal blnNewValue As Boolean)
  mudtVar.Checksum = blnNewValue
End Property
Public Property Get FromNumerator() As String
  FromNumerator = mstrFromNum
End Property
Public Property Let FromNumerator(ByVal strNewValue As String)
  mstrFromNum = strNewValue
  mudtVar.IsDirty = True
```

5

10

20

25

VBSCA -421-

End Property
Public Property Get FromDenominator() As String
FromDenominator = mstrFromDen
End Property
Public Property Let FromDenominator(ByVal strNewValue As String)
mstrFromDen = strNewValue mudtVar.IsDirty = True
End Property
Public Property Get ToNumerator() As String
ToNumerator = mstrToNum
End Property
Public Property Let ToNumerator(ByVal strNewValue As String)
mstrToNum = strNewValue mudtVar.IsDirty = True
End Property
Public Property Get ToDenominator() As String
ToDenominator = mstrToDen
End Property
Public Property Let ToDenominator(ByVal strNewValue As String)
mstrToDen = strNewValue mudtVar.IsDirty = True
End Property
Public Property Get ByNumerator() As String
ByNumerator = mstrByNum

```
End Property
        Public Property Let ByNumerator(ByVal strNewValue As String)
          mstrByNum = strNewValue
          mudtVar.IsDirty = True
        End Property
 5
        Public Property Get ByDenominator() As String
          ByDenominator = mstrByDen
        End Property
        Public Property Let ByDenominator(ByVal strNewValue As String)
10
          mstrByDen = strNewValue
          mudtVar.IsDirty = True
        End Property
 The about they also, they they
        Public Property Get MixedNumbers() As Boolean
          MixedNumbers = mblnMixedNumbers
        End Property
15
  IJ,
        Public Property Let MixedNumbers(ByVal blnNewValue As Boolean)
          mblnMixedNumbers = blnNewValue
          mudtVar.IsDirty = True
        End Property
20
       Public Property Get IsIndependent() As Boolean
          IsIndependent = mblnIsIndependent
        End Property
        Public Property Let IsIndependent(ByVal blnNewValue As Boolean)
          mblnIsIndependent = blnNewValue
          mudtVar.IsDirty = True
25
```

End Property

```
Public Sub Update(ByVal strName As String,
          ByVal strFromN As String, ByVal strFromD As String, _
          ByVal strToN As String, ByVal strToD As String,
 5
          ByVal strByN As String, ByVal strByD As String, _
          ByVal blnIsIndependent As Boolean, ByVal blnChecksum As Boolean,
          ByVal blnMixedNumber As Boolean)
          Variable Name = strName
          FromNumerator = strFromN
10
          From Denominator = str From D
          ToNumerator = strToN
          ToDenominator = strToD
          ByNumerator = strByN
15
          ByDenominator = strByD
          IsIndependent = blnIsIndependent
          Variable Checksum = blnChecksum
          MixedNumbers = blnMixedNumber
       End Sub
 Bin Ape from
       Public Function Variable_PrologFormat() As String
  Ü
          Dim str1 As String
25二
          If mblnIsIndependent Then
            str1 = "fraction(" & mudtVar.Name & "),offgrid(" & _
              mudtVar.Name & "),[" & _
              mstrFromNum & "/" & mstrFromDen & "<=" &
              mudtVar.Name & "<=" & mstrToNum & "/" & _
30
              mstrToDen & " step " & mstrByNum & "/" & mstrByDen & "]"
          Else
            str1 = "fraction(" & mudtVar.Name & ")"
          End If
35
          Variable PrologFormat = str1
       End Function
       Public Function Variable_ScreenFormat() As String
          Dim str1 As String
          Dim strOpt As String
```

```
If mudtVar.Checksum Then
             strOpt = "(C,"
           Else
             strOpt = "(c,"
 5
           End If
           If mblnMixedNumbers Then
             strOpt = strOpt & "M),"
10
           Else
             strOpt = strOpt \& "m),"
           End If
           If mblnIsIndependent Then
             str1 = mudtVar.Name & strOpt & ": Fraction, " &
15
               mstrFromNum & "/" & mstrFromDen & " to " & _
               mstrToNum & "/" & mstrToDen & " by " & _
               mstrByNum & "/" & mstrByDen
           Else
             str1 = mudtVar.Name & strOpt & ": Fraction"
           End If
  Ping offices (Ping offices Units Will)
           Variable ScreenFormat = str1
        End Function
25₂
        Public Property Get Variable ReadType(udtFile As File) As VariableType
  The Mar. It's
           Variable ReadType = mudtVar.ReadType(udtFile)
  <u></u>h
        End Property
        Public Sub Variable ReadObjectData(udtFile As File)
          Dim vField As Variant
30
          Call udtFile.ReadField(vField) ' reads version stamp
          Call udtFile.ReadField(vField)
          mudtVar.Name = vField
          Call udtFile.ReadField(vField)
          mudtVar.Enabled = vField
35
          Call udtFile.ReadField(vField)
          mudtVar.Checksum = vField
```

	Call udffile.Readfield(vfield) IsIndependent = vField
5	Call udtFile.ReadField(vField) FromNumerator = vField
10	Call udtFile.ReadField(vField) FromDenominator = vField
	Call udtFile.ReadField(vField) ToNumerator = vField
15	Call udtFile.ReadField(vField) ToDenominator = vField
	Call udtFile.ReadField(vField) ByNumerator = vField
2 6 1	Call udtFile.ReadField(vField) ByDenominator = vField
20 mg cm cm ca cm ca cm ca cm 25 mg cm ca	Call udtFile.ReadField(vField) MixedNumbers = vField
251	End Sub
ding drift fall in	Public Sub Variable_WriteObjectData(udtFile As File)
	Dim udtType As VariableType
	udtType = vtFraction
30 ^[]	Call udtFile.WriteField(udtType)
	Call udtFile.WriteField(mintVERSIONSTAMP)
	Call udtFile.WriteField(mudtVar.Name)
	Call udtFile.WriteField(mudtVar.Enabled)
2.5	Call udtFile.WriteField(mudtVar.Checksum)
35	Call udtFile.WriteField(IsIndependent)
	Call udtFile.WriteField(FromNumerator) Call udtFile.WriteField(FromDenominator)
	Call udtFile.WriteField(ToNumerator)
	Call udtFile.WriteField(ToDenominator)
40	Call udtFile.WriteField(ByNumerator)
	Call udtFile.WriteField(ByDenominator)
	Call udtFile.WriteField(MixedNumbers)

mudtVar.IsDirty = False

End Sub

' makes a copy of this object

Public Function Variable_Copy() As Variable

Dim udtVF As New VarFraction Dim udtV As Variable

Set udtV = udtVF

10

udtV.Name = mudtVar.Name udtV.Enabled = mudtVar.Index udtV.IsDirty = mudtVar.IsDirty udtV.Checksum = mudtVar.Checksum

15

udtVF.FromNumerator = FromNumerator udtVF.FromDenominator = FromDenominator udtVF.ByNumerator = ByNumerator udtVF.ByDenominator = ByDenominator udtVF.ToNumerator = ToNumerator udtVF.ToDenominator = ToDenominator udtVF.IsIndependent = IsIndependent udtVF.MixedNumbers = MixedNumbers

25

Set $Variable_Copy = udtV$

End Function

```
'Variable.cls
       VERSION 1.0 CLASS
       BEGIN
        MultiUse = 0 'False
 5
        Persistable = 0 'NotPersistable
        DataBindingBehavior = 0 'vbNone
        DataSourceBehavior = 0 'vbNone
        MTSTransactionMode = 0 'NotAnMTSObject
       END
       Attribute VB Name = "Variable"
10
       Attribute VB GlobalNameSpace = False
       Attribute VB Creatable = True
       Attribute VB PredeclaredId = False
       Attribute VB Exposed = False
       Attribute VB_Ext_KEY = "SavedWithClassBuilder", "Yes"
15
       Attribute VB Ext KEY = "Top Level", "Yes"
       Option Explicit
       Private mstrName As String
 ű
       Private mudtType As VariableType
       Private mlngIndex As Long
       Private mblnEnabled As Boolean
       Private mblnIsDirty As Boolean
       Private mblnChecksum As Boolean
       Public Enum VariableType
25
         vtInteger = 0
         vtReal = 1
         vtFraction = 2
         vtString = 3
         vtUntyped = 4
       End Enum
       Public Property Get name() As String
         name = mstrName
       End Property
       Public Property Let name(ByVal strNewValue As String)
35
         mstrName = strNewValue
           mblnIsDirty = True
```

```
End If
       End Property
       Public Property Get Typ() As VariableType
          Typ = mudtType
       End Property
       Public Property Let Typ(ByVal udtNewValue As VariableType)
         If mudtType 	<> udtNewValue Then
            mudtType = udtNewValue
10
            mblnIsDirty = True
          End If
       End Property
 Public Property Get index() As Long
         index = mlngIndex
End Property
       Public Property Let index(ByVal lngNewValue As Long)
         If mlngIndex <> lngNewValue Then
            mlngIndex = lngNewValue
            mblnIsDirty = True
          End If
       End Property
       Public Property Get Enabled() As Boolean
25
          Enabled = mblnEnabled
       End Property
       Public Property Let Enabled(ByVal blnNewValue As Boolean)
         If mblnEnabled 		○ blnNewValue Then
            mblnEnabled = blnNewValue
30
            mblnIsDirty = True
```

```
End If
       End Property
       Public Property Let IsDirty(ByVal blnNewValue As Boolean)
 5
          mblnIsDirty = blnNewValue
       End Property
       Public Property Get IsDirty() As Boolean
          IsDirty = mblnIsDirty
10
       End Property
       Public Property Let Checksum(ByVal blnNewValue As Boolean)
          If mblnChecksum \Leftrightarrow blnNewValue Then
            mblnChecksum = blnNewValue
            mblnIsDirty = True
          End If
       End Property
       Public Property Get Checksum() As Boolean
20
          Checksum = mblnChecksum
       End Property
       ' implemented in the subclasses of Variable
       Public Function PrologFormat() As String
       End Function
       ' implemented in the subclasses of Variable
       Public Function ScreenFormat() As String
30
       End Function
       ' implemented in the subclasses of Variable
```

End Sub

' implemented in the subclasses of Variable

Public Sub WriteObjectData(udtFile As File)

5 End Sub

Public Property Get ReadType(udtFile As File) As VariableType

Dim udtType As VariableType

Call udtFile.ReadField(udtType)

10 ReadType = udtType

End Property

' implemented in the subclasses of Variable

Public Function Copy() As Variable

End Function

15 mg (Ch ...) Sopring like and the mile and the like and

VBSCA -431-

	'VarInteger.cls
	VERSION 1.0 CLASS
	BEGIN
	MultiUse = -1 'True
5	END
	Attribute VB_Name = "VarInteger"
	Attribute VB_GlobalNameSpace = False
	Attribute VB_Creatable = True
	Attribute VB_PredeclaredId = False
10	Attribute VB_Exposed = False
	Option Explicit
	Implements Variable
	Private mudtVar As Variable
	' current version of data produced by this class
15	Const mintVERSIONSTAMP As Integer = 1
41 71	Private mstrFrom As String
117	Private mstrTo As String
12. 22.	Private mstrBy As String
15 The Paris of th	Private mblnIsIndependent As Boolean
	Private Sub Class_Initialize()
£ Ćì	Cat mudtVon - Novy Voniable
178-1778 1878 1874, 1774, 18 1838 1838 1878 1878 1838	Set mudtVar = New Variable
C)	End Sub
ļah 	End Sub
	Private Sub Class_Terminate()
Li	111/410 540 51455_1411111400()
25	Set mudtVar = Nothing
	· · · · · · · · · · · · · · · · · · ·
	End Sub
	'Delegated to Class Variable
	Public Property Get Variable Name() As String
30	
	Variable_Name = mudtVar.Name
	End Property
	Life I Topolty
	' Delegated to Class Variable

```
Public Property Let Variable Name(ByVal RHS As String)
          mudtVar.Name = RHS
        End Property
        ' Delegated to Class Variable
        Public Property Get Variable_Typ() As VariableType
 5
          Variable_Typ = mudtVar.Typ
        End Property
        ' Delegated to Class Variable
10
        Public Property Let Variable Typ(ByVal udtNewValue As VariableType)
          mudtVar.Typ = udtNewValue
        End Property
  Mar III
        ' Delegated to Class Variable
        Public Property Get Variable_Index() As Long
          Variable_Index = mudtVar.Index
        End Property
        'Delegated to Class Variable
        Public Property Let Variable Index(ByVal lngNewValue As Long)
          mudtVar.Index = lngNewValue
        End Property
        ' Delegated to Class Variable
        Public Property Get Variable Enabled() As Boolean
25
          Variable Enabled = mudtVar.Enabled
        End Property
        'Delegated to Class Variable
       Public Property Let Variable Enabled(ByVal RHS As Boolean)
          mudtVar.Enabled = RHS
```

```
End Property
       'Delegated to Class Variable
       Public Property Get Variable IsDirty() As Boolean
 5
          Variable IsDirty = mudtVar.IsDirty
       End Property
        ' Delegated to Class Variable
       Public Property Let Variable IsDirty(ByVal RHS As Boolean)
          mudtVar.IsDirty = RHS
10
       End Property
       ' Delegated to Class Variable
       Public Property Get Variable Checksum() As Boolean
Variable_Checksum = mudtVar.Checksum
       End Property
        ' Delegated to Class Variable
       Public Property Let Variable Checksum(ByVal blnNewValue As Boolean)
          mudtVar.Checksum = blnNewValue
       End Property
       Public Property Get From() As String
          From = mstrFrom
       End Property
       Public Property Let From(ByVal strNewValue As String)
25
          If mstrFrom 		 strNewValue Then
            mstrFrom = strNewValue
            mudtVar.IsDirty = True
          End If
```

End Property

```
Public Property Get Too() As String
         Too = mstrTo
       End Property
       Public Property Let Too(ByVal strNewValue As String)
         mstrTo = strNewValue
            mudtVar.IsDirty = True
         End If
       End Property
10
       Public Property Get By() As String
         By = mstrBy
 W. T.
       End Property
 O
       Public Property Let By(ByVal strNewValue As String)
 41
         If mstrBy ⇔ strNewValue Then
15
            mstrBy = strNewValue
 Ų,
           mudtVar.IsDirty = True
         End If
       End Property
 in h
       Public Property Get IsIndependent() As Boolean
20
         IsIndependent = mblnIsIndependent
       End Property
       Public Property Let IsIndependent(ByVal blnNewValue As Boolean)
         If mblnIsIndependent ⇔ blnNewValue Then
           mblnIsIndependent = blnNewValue
           mudtVar.IsDirty = True
25
         End If
       End Property
```

```
Public Sub Update(ByVal strName As String,
          ByVal strFrom As String, ByVal strTo As String, ByVal strBy As String, _
          ByVal blnIsIndependent As Boolean, ByVal blnChecksum As Boolean)
          Variable Name = strName
 5
          From = strFrom
          Too = strTo
          By = strBy
          IsIndependent = blnIsIndependent
10
          Variable Checksum = blnChecksum
        End Sub
        Public Function Variable PrologFormat() As String
          Dim str1 As String
15
          If mblnIsIndependent Then
            str1 = "int(" & mudtVar.Name & "),[" & mstrFrom & "<=" &
                 mudtVar.Name & "<=" & mstrTo & " step " & mstrBy & "]"
          Else
 g'n
201
            str1 = "int(" & mudtVar.Name & ")"
          End If
 Han alim Kim Alim
          Variable PrologFormat = str1
251
1
        End Function
        Public Function Variable ScreenFormat() As String
          Dim str1 As String
          Dim strT As String
          Dim strOpt As String
30
          If mudtVar.Checksum Then
            strOpt = "(C)"
          Else
            strOpt = "(c)"
          End If
35
          If mblnIsIndependent Then
            str1 = mudtVar.Name & strOpt & ": Int, " & mstrFrom & " to " &
               mstrTo & " by " & mstrBy
          Else
            str1 = mudtVar.Name & strOpt & ": Int"
40
```

```
Variable ScreenFormat = str1
        End Function
        Public Property Get Variable ReadType(udtFile As File) As VariableType
 5
          Variable ReadType = mudtVar.ReadType(udtFile)
        End Property
        Public Sub Variable ReadObjectData(udtFile As File)
          Dim vField As Variant
          Call udtFile.ReadField(vField) ' reads version stamp
10
          Call udtFile.ReadField(vField)
          mudtVar.Name = vField
  Ęģ.
  <u>G</u>1
1547
          Call udtFile.ReadField(vField)
  Henry Hern Henry Green
          mudtVar.Enabled = vField
          Call udtFile.ReadField(vField)
          mudtVar.Checksum = vField
          Call udtFile.ReadField(vField)
          From = vField
          Call udtFile.ReadField(vField)
          Too = vField
          Call udtFile.ReadField(vField)
          By = vField
30
          Call udtFile.ReadField(vField)
          IsIndependent = vField
        End Sub
        Public Sub Variable WriteObjectData(udtFile As File)
```

Dim udtType As VariableType

End If

```
udtType = vtInteger
          Call udtFile.WriteField(udtType)
         Call udtFile.WriteField(mintVERSIONSTAMP)
          Call udtFile.WriteField(mudtVar.Name)
          Call udtFile.WriteField(mudtVar.Enabled)
 5
          Call udtFile.WriteField(mudtVar.Checksum)
          Call udtFile.WriteField(From)
          Call udtFile.WriteField(Too)
          Call udtFile.WriteField(By)
          Call udtFile.WriteField(IsIndependent)
10
          mudtVar.IsDirty = False
       End Sub.
       ' makes a copy of this object
       Public Function Variable Copy() As Variable
15
          Dim udtVI As New VarInteger
          Dim udtV As Variable
 4)
 ۵ì
 U
          Set udtV = udtVI
20 🚉
 47
          udtV.Name = mudtVar.Name
          udtV.Typ = vtInteger
          udtV.Enabled = mudtVar.Index
          udtV.IsDirty = mudtVar.IsDirty
          udtV.Checksum = mudtVar.Checksum
          udtVI.From = From
          udtVI.Too = Too
          udtVI.By = By
          udtVI.IsIndependent = IsIndependent
          Set Variable Copy = udtV
```

End Function

	v arkear.cis
	VERSION 1.0 CLASS
	BEGIN
	MultiUse = -1 'True
5	END
	Attribute VB Name = "VarReal"
	Attribute VB GlobalNameSpace = False
	Attribute VB_Creatable = True
	Attribute VB PredeclaredId = False
10	Attribute VB Exposed = False
	Option Explicit
	- -
	Implements Variable
	Private mudtVar As Variable

	' current version of data produced by this class
15	Const mintVERSIONSTAMP As Integer = 2
15. I will will will will will will will wi	Private mstrFrom As String
U1	Private mstrTo As String
U I	Private mstrBy As String
rija Liji	Private mblnTrailingZeros As Boolean
202	Private mstrPrecision As String
11	Private mblnIsIndependent As Boolean
	Private mblnIsOnGrid As Boolean
	1 Tivate monisonoria 713 Boolean
4]	Private Sub Class Initialize()
4)	Titale du dias_imianze()
	Set mudtVar = New Variable
2 5	out made van 1000 vanaoio
L	End Sub
	Liid Suo
	Private Sub Class_Terminate()
	Trivate bab class_Terminate()
	Set mudtVar = Nothing
	Det made van Hommig
30	End Sub
50	Liid Sdo
	' Delegated to Class Variable
	Public Property Get Variable_Name() As String
	Table Troporty Get Tallable_Itallie() 713 Build
	Variable Name = mudtVar Name

End Property 'Delegated to Class Variable Public Property Let Variable_Name(ByVal RHS As String) mudtVar.Name = RHS**End Property** 'Delegated to Class Variable Public Property Get Variable Typ() As VariableType Variable Typ = mudtVar.Typ**End Property** 'Delegated to Class Variable Public Property Let Variable_Typ(ByVal udtNewValue As VariableType) mudtVar.Typ = udtNewValue**End Property** 'Delegated to Class Variable Public Property Get Variable Enabled() As Boolean Variable Enabled = mudtVar.Enabled **End Property** 'Delegated to Class Variable Public Property Let Variable_Enabled(ByVal RHS As Boolean) mudtVar.Enabled = RHS **End Property** 'Delegated to Class Variable

5

10

25 Public Property Get Variable_Index() As Long

Variable Index = mudtVar.Index

End Property

' Delegated to Class Variable 30

```
Public Property Let Variable Index(ByVal lngNewValue As Long)
  mudtVar.Index = lngNewValue
End Property
' Delegated to Class Variable
Public Property Get Variable IsDirty() As Boolean
  Variable IsDirty = mudtVar.IsDirty
End Property
' Delegated to Class Variable
Public Property Let Variable IsDirty(ByVal RHS As Boolean)
  mudtVar.IsDirty = RHS
End Property
' Delegated to Class Variable
Public Property Get Variable Checksum() As Boolean
  Variable_Checksum = mudtVar.Checksum
End Property
' Delegated to Class Variable
Public Property Let Variable Checksum(ByVal blnNewValue As Boolean)
  mudtVar.Checksum = blnNewValue
End Property
Public Property Get From() As String
  From = mstrFrom
End Property
Public Property Let From(ByVal strNewValue As String)
  mstrFrom = strNewValue
    mudtVar.IsDirty = True
```

5

10

25

```
End If
                               End Property
                               Public Property Get Too() As String
                                        Too = mstrTo
                               End Property
                               Public Property Let Too(ByVal strNewValue As String)
                                       If mstrTo <> strNewValue Then
                                                 mstrTo = strNewValue
                                                mudtVar.IsDirty = True
10
                                        End If
                              End Property
                              Public Property Get By() As String
      41
      T II
                                       By = mstrBy
      office Man office of the state 
                               End Property
15
                              Public Property Let By(ByVal strNewValue As String)
                                       If mstrBy <> strNewValue Then
                                                mstrBy = strNewValue
                                                mudtVar.IsDirty = True
                                        End If
                              End Property
                              Public Property Get TrailingZeros() As Boolean
                                        TrailingZeros = mblnTrailingZeros
                              End Property
                              Public Property Let TrailingZeros(ByVal blnNewValue As Boolean)
                                      If mblnTrailingZeros ⇔ blnNewValue Then
25
                                                mblnTrailingZeros = blnNewValue
                                                mudtVar.IsDirty = True
```

```
End If
        End Property
        Public Property Get IsOnGrid() As Boolean
          IsOnGrid = mblnIsOnGrid
        End Property
        Public Property Let IsOnGrid(ByVal blnNewValue As Boolean)
          If mblnIsOnGrid >> blnNewValue Then
            mblnIsOnGrid = blnNewValue
            mudtVar.IsDirty = True
          End If
10
        End Property
 Public Property Get Precision() As String
  ជា
          Precision = mstrPrecision
        End Property
151
        Public Property Let Precision(ByVal strNewValue As String)
          If mstrPrecision \Leftrightarrow strNewValue Then
            mstrPrecision = strNewValue
            mudtVar.IsDirty = True
          End If
        End Property
        Public Property Get DecimalPlaces() As Integer
          If InStr(1, mstrPrecision, ".") = 0 Then
            DecimalPlaces = 0
          Else
            DecimalPlaces = Len(mstrPrecision) - 1
25
          End If
        End Property
        Public Property Get IsIndependent() As Boolean
```

```
IsIndependent = mblnIsIndependent
        End Property
        Public Property Let IsIndependent(ByVal blnNewValue As Boolean)
          If mblnIsIndependent ⇔ blnNewValue Then
            mblnIsIndependent = blnNewValue
 5
            mudtVar.IsDirty = True
          End If
        End Property
        Public Sub Update(ByVal strName As String,
10
          ByVal strFrom As String, ByVal strTo As String, ByVal strBy As String,
          ByVal blnIsIndependent As Boolean, ByVal blnChecksum As Boolean, _
          ByVal blnTrailingZeros As Boolean,
          ByVal strPrecision As String, ByVal blnIsOnGrid As Boolean)
          Variable Name = strName
          From = strFrom
          Too = strTo
 frie den Krii
          By = strBy
          IsIndependent = blnIsIndependent
20 ===
          Variable Checksum = blnChecksum
 IJ
          TrailingZeros = blnTrailingZeros
          Precision = strPrecision
          IsOnGrid = blnIsOnGrid
        End Sub
        Public Function Variable PrologFormat() As String
          Dim str1 As String
30
          If mblnIsIndependent Then
            str1 = "real({" & mudtVar.Name & "," & mstrPrecision & "}),[" _
              & mstrFrom & "<=" & mudtVar.Name & "<=" & mstrTo & " step " &
              mstrBy & "]"
          Else
            str1 = "real(" & mudtVar.Name & ")"
35
          End If
          If Not mblnIsOnGrid Then
            str1 = str1 & ",offgrid(" & mudtVar.Name & ")"
```

```
End If
          Variable PrologFormat = str1
        End Function
 5
        Public Function Variable ScreenFormat() As String
          Dim str1 As String
          Dim strOpt As String
          If mudtVar.Checksum Then
10
            strOpt = "(C,"
          Else
            strOpt = "(c,"
          End If
15
          If mblnTrailingZeros Then
            strOpt = strOpt & "T,"
          Else
 IJ.
            strOpt = strOpt & "t,"
 Ø
207
          End If
 j.
          If mblnIsOnGrid Then
            strOpt = strOpt & "G,"
 £
          Else
            strOpt = strOpt & "g,"
          End If
          strOpt = strOpt & mstrPrecision & ")"
          If mblnIsIndependent Then
            str1 = mudtVar.Name & strOpt & ": Real, " & mstrFrom & " to " & _
               mstrTo & " by " & mstrBy
            str1 = mudtVar.Name & strOpt & ": Real"
35
          End If
          Variable ScreenFormat = str1
        End Function
       Public Property Get Variable_ReadType(udtFile As File) As VariableType
          Variable ReadType = mudtVar.ReadType(udtFile)
```

VBSCA -445-

	Public Sub Variable_ReadObjectData(udtFile As File)
5	Dim vField As Variant Dim intVersion As Integer
3	Call udtFile.ReadField(vField) ' reads version stamp intVersion = vField
10	Call udtFile.ReadField(vField) mudtVar.Name = vField
	Call udtFile.ReadField(vField) mudtVar.Enabled = vField
15	Call udtFile.ReadField(vField) mudtVar.Checksum = vField
Harth Storte, If the	Call udtFile.ReadField(vField) From = vField
20 mg	Call udtFile.ReadField(vField) Too = vField
25 1	Call udtFile.ReadField(vField) By = vField
25	Call udtFile.ReadField(vField) TrailingZeros = vField
3 (= 3	Call udtFile.ReadField(vField) Precision = vField
2.5	Call udtFile.ReadField(vField) IsIndependent = vField
35	If intVersion < 2 Then ' this field is new to version 2 of VarReal IsOnGrid = True
40	Else Call udtFile.ReadField(vField) IsOnGrid = vField End If

End Property

End Sub

Public Sub Variable WriteObjectData(udtFile As File)

Dim udtType As VariableType

- udtType = vtReal

 Call udtFile.WriteField(udtType)
 - Call udtFile.WriteField(mintVERSIONSTAMP)
 - Call udtFile.WriteField(mudtVar.Name)
 - Call udtFile.WriteField(mudtVar.Enabled)
 - Call udtFile.WriteField(mudtVar.Checksum)
- 10 Call udtFile.WriteField(From)
 - Call udtFile.WriteField(Too)
 - Call udtFile.WriteField(By)
 - Call udtFile.WriteField(TrailingZeros)
 - Call udtFile.WriteField(Precision)
- 15 Call udtFile.WriteField(IsIndependent)
 - Call udtFile.WriteField(IsOnGrid)

mudtVar.IsDirty = False

End Sub

ĽĴ

đ1

30

35

40

'makes a copy of this object

Public Function Variable_Copy() As Variable

Dim udtVR As New VarReal

Dim udtV As Variable

 25^{1} Set udtV = udtVR

udtV.Name = mudtVar.Name

udtV.Typ = vtReal

udtV.Enabled = mudtVar.Index

udtV.IsDirty = mudtVar.IsDirty

udtV.Checksum = mudtVar.Checksum

udtVR.From = From

udtVR.Too = Too

udtVR.By = By

udtVR.Precision = Precision

udtVR.TrailingZeros = TrailingZeros

udtVR.IsIndependent = IsIndependent

udtVR.IsOnGrid = IsOnGrid

Set Variable Copy = udtV

VBSCA -448-

	vaisting.cis
	VERSION 1.0 CLASS
	BEGIN
	MultiUse = -1 'True
5	END
	Attribute VB_Name = "VarString"
	Attribute VB GlobalNameSpace = False
	Attribute VB Creatable = True
	Attribute VB PredeclaredId = False
10	Attribute VB_Exposed = False
	Option Explicit
	Gruen Zirpinen
	Implements Variable
	Private mudtVar As Variable
	'current version of data produced by this class
15	Const mintVERSIONSTAMP As Integer = 1
Ę)	<u> </u>
W.	Private mstrDelimiter As String
414 414	3
W I ·	Private mblnIsIndexed As Boolean
.:T	
15. The sease seas	Private mcolString As New Collection
47	•
	Private Sub Class_Initialize()
20	Set mudtVar = New Variable
M	
	End Sub
tad prij	
ģens)	Private Sub Class_Terminate()
	·
	Set mudtVar = Nothing
25	
	End Sub
	' Delegated to Class Variable
	Public Property Get Variable_Name() As String
30	Variable_Name = mudtVar.Name
	End Property

' Delegated to Class Variable Public Property Let Variable Name(ByVal RHS As String) mudtVar.Name = RHS**End Property** 'Delegated to Class Variable 5 Public Property Get Variable Typ() As Variable Type Variable Typ = mudtVar.Typ**End Property** 10 'Delegated to Class Variable Public Property Let Variable_Typ(ByVal udtNewValue As VariableType) mudtVar.Typ = udtNewValue **End Property** Ţ, Œ١ ' Delegated to Class Variable Ų٦ 15 Public Property Get Variable Index() As Long Man all a Man Variable Index = mudtVar.Index **End Property** ' Delegated to Class Variable Public Property Let Variable_Index(ByVal lngNewValue As Long) mudtVar.Index = lngNewValue **End Property** ' Delegated to Class Variable 25 Public Property Get Variable Enabled() As Boolean Variable Enabled = mudtVar.Enabled **End Property** ' Delegated to Class Variable Public Property Let Variable_Enabled(ByVal RHS As Boolean)

End Property ' Delegated to Class Variable 5 Public Property Get Variable_IsDirty() As Boolean Variable IsDirty = mudtVar.IsDirty **End Property** ' Delegated to Class Variable Public Property Let Variable IsDirty(ByVal RHS As Boolean) 10 mudtVar.IsDirty = RHS**End Property** ' Delegated to Class Variable Public Property Get Variable Checksum() As Boolean Ō 15 Hope and and the Variable Checksum = mudtVar.Checksum **End Property** ' Delegated to Class Variable Public Property Let Variable Checksum(ByVal blnNewValue As Boolean) mudtVar.Checksum = blnNewValue **End Property** Public Property Get Delimiter() As String Delimiter = mstrDelimiter 25 **End Property** Public Property Let Delimiter(ByVal strNewValue As String) If mstrDelimiter <> strNewValue Then mstrDelimiter = strNewValue 30 mudtVar.IsDirty = True

mudtVar.Enabled = RHS

End If

		End Property
)		Public Property Get IsIndexed() As Boolean
		IsIndexed = mblnIsIndexed
	5	End Property
)		Public Property Let IsIndexed(ByVal blnNewValue As Boolean)
		mblnIsIndexed = blnNewValue
•		End Property
	10	Public Property Get StringCollection() As Collection
		Set StringCollection = mcolString
)		End Property
		Public Property Let StringCollection(ByVal colNewValue As Collection)
	4 7 1 5 5	Dim intIndex As Integer
,	7. (1974) 1970, 19	If mcolString.Count <> colNewValue.Count Then
	£ .	Set mcolString = colNewValue mudtVar.IsDirty = True
)	20-1 11 11	Exit Property End If
		For intIndex = 1 To mcolString.Count If mcolString.Item(intIndex) \Leftrightarrow colNewValue.Item(intIndex) Then Set mcolString = colNewValue
)		mudtVar.IsDirty = True Exit Property
		End If Next intIndex
	30	End Property
•		
		'returns the largest number of delimited substrings in the string collection Public Property Get NumIndices() As Integer
)	35	Dim intD As Integer

```
Dim intI As Integer
          Dim varS As Variant
          Dim udtSubStr As New SubString
 5
          'if there are no strings in the collection
          If mcolString.Count = 0 Then
             NumIndices = 1
            Exit Property
          End If
10
          udtSubStr.Delimiter = mstrDelimiter
          For Each varS In mcolString
             udtSubStr.StringValue = varS
            intD = udtSubStr.NumSubStrings
15
            If intD > intHiD Then
               intHiD = intD
            End If
          Next varS
NumIndices = intHiD
        End Property
        Public Function Variable PrologFormat() As String
25
          Variable PrologFormat = ""
        End Function
        Public Function Variable ScreenFormat() As String
  Œ)
  Dim str1 As String
          Dim strS As String
30
          Dim intIndex As Integer
          Dim strOpt As String
          If mudtVar.Checksum Then
            strOpt = "(C,"
35
          Else
            strOpt = "(c,"
          End If
          strOpt = strOpt & Str(NumIndices) & "," & mstrDelimiter & ")"
40
```

```
For intIndex = 1 \text{ To } 3
             If mcolString.Count >= intIndex Then
               strS = strS & mcolString.Item(intIndex)
               If mcolString.Count > intIndex Then
 5
                 strS = strS \& ","
               End If
             End If
10
          Next intIndex
          If mcolString.Count > 3 Then
             strS = strS & "..."
          End If
15
          str1 = mudtVar.Name & strOpt & ": String, in [" & strS & "]"
          Variable_ScreenFormat = str1
 C).
        End Function
 Ü.
 Ø١
20.
        Public Property Get Variable ReadType(udtFile As File) As VariableType
 Man allow
          Variable ReadType = mudtVar.ReadType(udtFile)
        End Property
        Public Sub Variable ReadObjectData(udtFile As File)
          Dim vField As Variant
          Dim intCount As Integer
          Call udtFile.ReadField(vField) ' reads version stamp
          Call udtFile.ReadField(vField)
          mudtVar.Name = vField
30
          Call udtFile.ReadField(vField)
          mudtVar.Enabled = vField
          Call udtFile.ReadField(vField)
          mudtVar.Checksum = vField
35
          Call udtFile.ReadField(vField)
          mstrDelimiter = vField
```

	Call udtFile.ReadField(vField) mblnIsIndexed = vField
	Call udtFile.ReadField(vField) intCount = vField
5	Dim intI As Integer
10	' read in the strings For intI = 1 To intCount
10	Call udtFile.ReadField(vField) Call mcolString.Add(vField)
15	Next intI
15	End Sub
	Public Sub Variable_WriteObjectData(udtFile As File)
	Dim udtType As VariableType
20 mg	udtType = vtString Call udtFile.WriteField(udtType) Call udtFile.WriteField(mintVERSIONSTAMP) Call udtFile.WriteField(mudtVar.Name) Call udtFile.WriteField(mudtVar.Enabled) Call udtFile.WriteField(mudtVar.Checksum)
	Call udtFile.WriteField(mstrDelimiter) Call udtFile.WriteField(mblnIsIndexed)
25 The Late of the second seco	Dim intCount As Integer
	<pre>intCount = mcolString.Count Call udtFile.WriteField(intCount)</pre>
25	Dim intI As Integer
35	' write out the strings For intI = 1 To mcolString.Count Call udtFile.WriteField(mcolString.Item(intI)) Next intI
40	mudtVar.IsDirty = False

End Sub

•		' makes a copy of this object Public Function Variable_Copy() As Variable
	5	Dim udtVS As New VarString Dim udtV As Variable Dim varS As Variant
		Set $udtV = udtVS$
1	10	udtV.Name = mudtVar.Name udtV.Typ = vtString udtV.Enabled = mudtVar.Index udtV.IsDirty = mudtVar.IsDirty
	1.5	udtV.Checksum = mudtVar.Checksum
	15	udtVS.Delimiter = Delimiter udtVS.IsIndexed = IsIndexed
	<u>0</u>	Set Variable_Copy = udtV
	20	For Each varS In mcolString Call udtVS.StringCollection.Add(varS) Next varS
,	25	End Function
		' VarUntyped.cls
÷		VERSION 1.0 CLASS BEGIN
	30	MultiUse = -1 'True END
		Attribute VB_Name = "VarUntyped" Attribute VB_GlobalNameSpace = False Attribute VB_Creatable = True
	35	Attribute VB_PredeclaredId = False Attribute VB_Exposed = False Option Explicit

Implements Variable

Private mudtVar As Variable 'current version of data produced by this class Const mintVERSIONSTAMP As Integer = 1 Private Sub Class_Initialize() Set mudtVar = New Variable 5 End Sub Private Sub Class Terminate() Set mudtVar = Nothing10 End Sub ' Delegated to Class Variable Public Property Get Variable_Name() As String 15 Variable Name = mudtVar.Name Him then his Am Am **End Property** 'Delegated to Class Variable Public Property Let Variable_Name(ByVal RHS As String) mudtVar.Name = RHS **End Property** ' Delegated to Class Variable Public Property Get Variable Typ() As Variable Type Variable Typ = mudtVar.Typ25 **End Property** 'Delegated to Class Variable Public Property Let Variable Typ(ByVal udtNewValue As VariableType) mudtVar.Typ = udtNewValue

End Property

'Delegated to Class Variable Public Property Get Variable_Index() As Long Variable Index = mudtVar.Index**End Property** 'Delegated to Class Variable Public Property Let Variable Index(ByVal lngNewValue As Long) mudtVar.Index = lngNewValue **End Property** ' Delegated to Class Variable Public Property Get Variable_Enabled() As Boolean Variable Enabled = mudtVar.Enabled **End Property** 'Delegated to Class Variable Public Property Let Variable Enabled(ByVal RHS As Boolean) mudtVar.Enabled = RHS **End Property** 'Delegated to Class Variable Public Property Get Variable IsDirty() As Boolean Variable IsDirty = mudtVar.IsDirty **End Property** 'Delegated to Class Variable Public Property Let Variable IsDirty(ByVal RHS As Boolean) mudtVar.IsDirty = RHS**End Property**

5

10

ወ1 1**5**ያገ

Min Man Min

25

30

' Delegated to Class Variable

Public Property Get Variable Checksum() As Boolean

```
Variable Checksum = mudtVar.Checksum
       End Property
       ' Delegated to Class Variable
 5
       Public Property Let Variable Checksum(ByVal blnNewValue As Boolean)
          mudtVar.Checksum = blnNewValue
       End Property
       Public Function Variable PrologFormat() As String
10
          Variable PrologFormat = ""
       End Function
       Public Function Variable_ScreenFormat() As String
         Dim str1 As String
         Dim strS As String
         Dim intIndex As Integer
         Dim strOpt As String
         If mudtVar.Checksum Then
            strOpt = "(C)"
         Else
            strOpt = "(c)"
         End If
         str1 = mudtVar.Name & strOpt & ": Untyped"
          Variable ScreenFormat = str1
       End Function
30
       Public Property Get Variable ReadType(udtFile As File) As VariableType
         Variable ReadType = mudtVar.ReadType(udtFile)
       End Property
       Public Sub Variable ReadObjectData(udtFile As File)
         Dim vField As Variant
```

	Dim intCount As Integer
5	Call udtFile.ReadField(vField) ' reads version stamp Call udtFile.ReadField(vField) mudtVar.Name = vField
	Call udtFile.ReadField(vField) mudtVar.Enabled = vField
10	Call udtFile.ReadField(vField) mudtVar.Checksum = vField
	End Sub
	Public Sub Variable_WriteObjectData(udtFile As File)
15	Dim udtType As VariableType
20	udtType = vtUntyped Call udtFile.WriteField(udtType) Call udtFile.WriteField(mintVERSIONSTAMP) Call udtFile.WriteField(mudtVar.Name) Call udtFile.WriteField(mudtVar.Enabled) Call udtFile.WriteField(mudtVar.Checksum)
#	mudtVar.IsDirty = False
2 5	End Sub
25-12-13-13-13-13-13-13-13-13-13-13-13-13-13-	' makes a copy of this object Public Function Variable_Copy() As Variable
i i	Dim udtV As New Variable
30	udtV.Name = mudtVar.Name udtV.Typ = vtUntyped udtV.Enabled = mudtVar.Index udtV.IsDirty = mudtVar.IsDirty udtV.Checksum = mudtVar.Checksum

End Function

Set Variable_Copy = udtV

VBSCA -461-

	' Win32API.cls VERSION 1.0 CLASS BEGIN
5	MultiUse = -1 'True END Attribute VB Name = "Win32API"
10	Attribute VB_GlobalNameSpace = False Attribute VB_Creatable = True Attribute VB_PredeclaredId = False Attribute VB_Exposed = False ' used for making calls to the Win32 API Option Explicit
15	Private Type FILETIME dwLowDateTime As Long dwHighDateTime As Long End Type
in in	Private Const MAX_PATH = 260
	Private Type WIN32_FIND_DATA dwFileAttributes As Long ftCreationTime As FILETIME ftLastAccessTime As FILETIME ftLastWriteTime As FILETIME nFileSizeHigh As Long nFileSizeLow As Long dwReserved0 As Long dwReserved1 As Long cFileName As String * MAX_PATH cAlternate As String * 14 End Type
30	Private Const INVALID_HANDLE_VALUE = -1
	Private Declare Function FindFirstFile Lib "kernel32" Alias "FindFirstFileA" _ (ByVal lpFileName As String, lpFindFileData As WIN32_FIND_DATA) As Long
35	Private Declare Function FindNextFile Lib "kernel32" Alias "FindNextFileA" (ByVal hFileName As Long, lpFindFileData As WIN32_FIND_DATA) As Long
	Private Declare Function FindClose Lib "kernel32" (ByVal hFindFile As Long) As Long
	Private Declare Function GetCurrentDirectory Lib "kernel32" _
	VBSCA -462-

```
Alias "GetCurrentDirectoryA" (ByVal nBufferLength As Long,
         ByVal lpBuffer As String) As Long
       Private Declare Function SendMessageLong Lib "user32" Alias "SendMessageA"
         (ByVal hwnd As Long,
 5
         ByVal Msg As Long, _
         ByVal wParam As Long, _
         ByVal lParam As Long) As Long
       Private Declare Function SystemParametersInfo Lib "user32" _
10
         Alias "SystemParametersInfoA" (ByVal uAction As Long,
         ByVal uParam As Long, ByRef lpvParam As Any,
         ByVal fuWinIni As Long) As Long
       Private Const SPI_GETDRAGFULLWINDOWS = 38
15
       Private Const SPI SETDRAGFULLWINDOWS = 37
       Private Const SPIF SENDWININICHANGE = 2
       Public Function IsFullWindowDragOn() As Boolean
 ű
         Dim result As Long
 m
 Li
         'Call API and check for successful call.
20҈
         If SystemParametersInfo(SPI GETDRAGFULLWINDOWS, 0&, result, 0&) \Leftrightarrow 0 Then
 ű
           'Feature supported now check value of result.
           If result = 0 Then
              IsFullWindowDragOn = False
2<del>5</del>-1
              IsFullWindowDragOn = True
           End If
           'Call failed, feature not supported.
           IsFullWindowDragOn = False
         End If
       End Function
       Public Sub TurnOffFullWindowDrag()
         Dim result As Long
35
         result = SystemParametersInfo(SPI SETDRAGFULLWINDOWS, 0&, _
           ByVal vbNullString, SPIF SENDWININICHANGE)
       End Sub
```

	Public Sub TurnOnFullWindowDrag()
	Dim result As Long
5	result = SystemParametersInfo(SPI_SETDRAGFULLWINDOWS, 1&, ByVal vbNullString, SPIF_SENDWININICHANGE)
	End Sub
	' returns true if strFN exists Public Function FileExists(ByVal strFN) As Boolean
10	Dim lngHandle As Long Dim w32FindData As WIN32_FIND_DATA
	lngHandle = FindFirstFile(strFN, w32FindData)
15. The second s	If lngHandle = INVALID_HANDLE_VALUE Then FileExists = False Else FileExists = True Call FindClose(lngHandle) End If
## ## ### ## ### ###	End Function
	' returns a collection of file names that satisfy strMask. The path seems to ' disappear from the returned file names.
	Public Function FindAllFiles(ByVal strMask As String) As Collection
man design	Dim lngHandle As Long Dim lngRet As Long
25	Dim w32FindData As WIN32_FIND_DATA Dim strFN As String Dim varI As Variant Dim colFNs As New Collection
20	lngHandle = FindFirstFile(strMask, w32FindData)
30	If lngHandle = INVALID_HANDLE_VALUE Then Exit Function End If

Do

	strFN = TrimAtFirstNull(w32FindData.cFileName) Call colFNs.Add(strFN) ' add to the collection
_	Loop Until FindNextFile(lngHandle, w32FindData) = 0
5	Set FindAllFiles = colFNs
	End Function
10	' returns the current directory Public Function CurrentDirectory() As String
	Dim strBuf As String Dim lngRet As Long Dim varI As Variant
15	strBuf = Space(300) lngRet = GetCurrentDirectory(300, strBuf) CurrentDirectory = TrimAtFirstNull(strBuf)
	End Function
20 10% 10% 10% 10% 10% 10% 10% 10% 10% 10	' enable full row select in list view control Public Sub EnableListViewFullRowSelect(lvwLV As ListView)
	Dim lngStyle As Long Dim lngL As Long
# ## ## ## ###########################	'get the current ListView style lngStyle = SendMessageLong(lvwLV.hwnd, LVM_GETEXTENDEDLISTVIEWSTYLE, 0& 0&)
	'set the extended style bit lngStyle = lngStyle Or LVS_EX_FULLROWSELECT
30	'set the new ListView style lngL = SendMessageLong(lvwLV.hwnd, LVM_SETEXTENDEDLISTVIEWSTYLE, 0&, lngStyle)

End Sub

	' Word.cls
	VERSION 1.0 CLASS
	BEGIN
	MultiUse = -1 'True
5	END
	Attribute VB_Name = "MSWord"
	Attribute VB_GlobalNameSpace = False
	Attribute VB_Creatable = True
	Attribute VB_PredeclaredId = False
10	Attribute VB_Exposed = False
	Option Explicit
	Private Const WM_CLOSE = &H10
	Private mWDApp As Word.Application
	Private Type RECT
15	left As Long
_{[=1}	top As Long
und Lift	right As Long
e e	bottom As Long
7 7 5 7 4 7 1 5 1 1 2 5 1 1 2 2 2 2 2 2 2 1 1 1 1 1	End Type
=†= 20≊1	Private Declare Function GetParent Lib "user32"
32	(ByVal hWndChild As Long) As Long
43	(=) · · · · · · · · · · · · · · · · · ·
B	Private Declare Function SetParent Lib "user32"
	(ByVal hWndChild As Long, ByVal hWndNewParent As Long) As Long
L)	
	Private Declare Function FindWindow Lib "user32"
25-	Alias "FindWindowA" (ByVal lpClassName As String,
e.	ByVal lpWindowName As String) As Long
	Private Declare Function SendMessage Lib "user32" _
	Alias "SendMessageA"
	(ByVal hwnd As Long, ByVal wMsg As Long, _
30	ByVal wParam As Long, lParam As Any) As Long
30	by var wraram As Long, ir aram As Amy) As Long
	Private Declare Function GetWindowRect Lib "user32"_
	(ByVal hwnd As Long, lpRect As RECT) As Long
	Private Declare Function SetWindowPos Lib "user32" _
	(ByVal hwnd As Long, ByVal hWndInsertAfter As Long, _
35	ByVal X As Long, ByVal Y As Long, ByVal cx As Long,
	ByVal cy As Long, ByVal wFlags As Long) As Long

```
Dim mlngHandle As Long
                         Dim origParent As Long
                         Dim origLeft As Long
                         Dim origTop As Long
    5
                         Dim origWidth As Long
                         Dim origHeight As Long
                         Private Sub Class Initialize()
                         ' mlngHandle = FindWindow("OpusApp", vbNullString)
                               Do While mlngHandle \Leftrightarrow 0
                                       SendMessage mlngHandle, WM CLOSE, mlngHandle, 0
10
                                       mlngHandle = FindWindow("OpusApp", vbNullString)
                            Loop
                                mlngHandle = FindWindow("OpusApp", vbNullString)
                                If mlngHandle \Leftrightarrow 0 Then
                                        Set mWDApp = GetObject(, "Word.Application.8")
                                Else
     đ
                                        On Error Resume Next
     The first of the state of the s
                                        Set mWDApp = GetObject(, "Word.Application.8")
                                       If err.Number = 0 Then
201
                                              MsgBox "Phantom WinWord detected!"
                                              Call mWDApp.Quit(False)
                                       Else
                                               err.Clear
                                       End If
                                        Set mWDApp = CreateObject("Word.Application.8")
                                End If
                                mlngHandle = FindWindow("OpusApp", vbNullString)
                                If mlngHandle 

○ 0 Then
                                       origParent = GetParent(mlngHandle)
30
                                       If mWDApp.left < 0 Then
                                             origLeft = 0
                                       Else
                                             origLeft = mWDApp.left
                                       End If
```

```
If mWDApp.top < 0 Then
             origTop = 0
            Else
             origTop = mWDApp.top
 5
            End If
            origWidth = mWDApp.Width
            origHeight = mWDApp.Height
            Call SetParent(mlngHandle, frmTCA.fraWord.hwnd)
         End If
10
         mWDApp.Visible = True
       End Sub
       Private Sub Class Terminate()
         mWDApp.Visible = False
         Call SetParent(mlngHandle, origParent)
         Call mWDApp.Move(origLeft, origTop)
         Call mWDApp.Resize(origWidth, origHeight)
         Call mWDApp.Quit(False) 'don't save!
       End Sub
       Public Property Get WordApp() As Word.Application
         Set WordApp = mWDApp
       End Property
       Public Property Get DocumentsCount() As Long
         DocumentsCount = mWDApp.Documents.Count
       End Property
25
       Public Property Get SelectionType() As Long
         SelectionType = mWDApp.Selection.Type
      · End Property
```

```
Public Property Get SelectionText() As String
  SelectionText = mWDApp.Selection.Text
End Property
Public Sub Resize()
  Dim WindowRect As RECT
  GetWindowRect frmTCA.fraWord.hwnd, WindowRect
  Dim lngH As Long
  Dim lngW As Long
  lngW = frmTCA.ScaleX(WindowRect.right - WindowRect.left, vbPixels, vbPoints)
  lngH = frmTCA.ScaleY(WindowRect.bottom - WindowRect.top, vbPixels, vbPoints)
  Call mWDApp.Resize(lngW, lngH)
  Call mWDApp.Move(0, 0)
  SetWindowPos mlngHandle, 0, 0, 0, _
    WindowRect.right - WindowRect.left,
     WindowRect.bottom - WindowRect.top, 64
  CommandBars("File").Controls("Exit").Enabled = False
End Sub
Public Sub CloseAllDocs()
  Dim docD As Document
  For Each docD In mWDApp.Documents
    If Not docD.ReadOnly Then
      docD.Close
    Else
      Call docD.Close(False)
    End If
  Next docD
```

10

20

End Sub